Perangkat Lunak HMI Untuk Sistem Supervisory Control Pada Pilot Plant Biodiesel

Endang Suryawati
P2 Informatika-LIPI
endang@informatika.lipi.go.id

Rika Sustika
P2 Informatika-LIPI
rika@informatika.lipi.go.id

Abstrak

Dalam suatu sistem supervisory control diperlukan perangkat lunak antar muka yang menjadi penghubung antara manusia (operator) dengan mesin atau peralatan yang dikendalikan. Perangkat lunak tersebut umumnya disebut sebagai HMI (Human Machine Interface) yang berupa Graphical User Interface (GUI) berbasis komputer. Makalah ini menjelaskan tentang pengembangan perangkat lunak HMI untuk sistem kontrol dan monitoring jarak jauh pada pilot plant biodiesel (metil ester) berbasis TCP/IP. Penggunaan format penyimpanan data standar (teknologi XML) dilakukan untuk kemudahan penggunaan koleksi data HMI oleh sub sistem lain. Metodologi pengembangan perangkat lunak mengikuti pendekatan Object Oriented Software Engineering (OOSE) dengan menggunakan notasi UML (Unified Modeling Language). Pengkodean dalam tahap implementasi dilakukan dengan menggunakan bahasa pemrograman Java untuk pengurangan aspek ketergantungan pada produk-produk berlisensi. Dari hasil kegiatan diperoleh sebuah perangkat lunak HMI berbasis open source yang berfungsi dalam hal pengawasan, pengendalian, dan pendukung otomatisasi proses pada pilot plant biodiesel.

Kata kunci: GUI, HMI, Java, Open Source, OOSE, Supervisory Control, TCP/IP, UML, XML

1. Pendahuluan

Sistem *supervisory control* adalah suatu sistem yang terdiri dari perangkat keras dan perangkat lunak yang diperlukan untuk melakukan pemantauan dan pengendalian terhadap suatu mesin atau *plant*, sehingga proses pada *plant* dapat berjalan sesuai prosedur yang berlaku.

Secara umum bagian-bagian yang membentuk suatu sistem *supervisory control* adalah:

- RTU (*Remote Terminal Unit*)
- Sensor dan aktuator (*field devices*)
- HMI (*Human Machine Interface*)

RTU merupakan pengendali utama dari jalannya proses pada suatu *plant*. RTU dapat berupa PLC (*Programmable Logic Controller*), PC (*Personal Computer*), PAC (*Programmable Automatic Controller*)[7] dan lain-lain.

Sensor terpasang sebagai input *plant* untuk dipantau setiap perubahan nilai-nilainya oleh RTU. Aktuator terpasang sebagai output *plant* untuk dikendalikan RTU pada kondisi nilai tertentu dari sensor. HMI merupakan perangkat lunak antar muka yang menjadi penghubung antara operator dengan *plant* yang dikendalikan. RTU secara berkala mengirimkan data kondisi input dan output pada *plant* ke HMI sesuai *request* dari HMI.

Perangkat lunak HMI dapat diidentikkan sebagai gambaran proses suatu sistem atau *plant* yang menampilkan kondisi input dan output *plant.* Perubahan jenis aplikasi yang dipantau dan dikendalikan oleh sistem, akan menyebabkan perubahan pada:

- Pemilihan jenis input dan output pada *plant.*
- Program yang terpasang dalam RTU.
- Pengaturan (*setting*) untuk menggunakan input dan ouput *plant*,

prosedur pengambilan data, prosedur pemilahan dan penyimpanan data.

Kondisi seperti ini dapat mendasari adanya ketergantungan HMI dengan sub-sub sistem lainnya dalam suatu *supervisory control*. Bentuk ketergantungan ini dapat ditemui dengan seringnya ditemukan produk HMI yang selalu terpadu dengan sistem RTU dalam suatu paket berlisensi. Permasalahan seputar formasi data yang harus disepakati oleh RTU dan HMI juga turut menunjang ketergantungan di antara kedua belah pihak tersebut. Adanya keterpaduan antara RTU dan HMI, tentunya ditujukan untuk memudahkan dalam melakukan perubahan baik pada sisi HMI maupun pada sisi RTU, ketika terjadi perubahan pada aplikasi yang dipantaunya.

Selama perubahan jenis aplikasi belum menuntut suatu perubahan yang menimbulkan kendala dalam pengembangan selanjutnya dari suatu sistem *supervisory* secara keseluruhan, maka penggunaan RTU dan HMI dalam satu paket berlisensi menjadi tidak bermasalah.

Penggunaan paket berlisensi tersebut akan menjadi masalah ketika kita tidak dapat melakukan pembaharuan baik pada sisi HMI maupun sisi RTU, untuk keperluan perubahan jenis aplikasi dan pengembangan sistem kita selanjutnya. Penawaran solusi dalam bentuk pembelian paket tambahan dengan *features* baru akan menjadi masalah ketika ada tuntutan terhadap penekanan biaya.

Kemandirian dalam hal mengembangkan RTU atau HMI, dapat menjadi salah satu solusi untuk mengurangi ketergantungan tersebut. Pengembangan perangkat lunak HMI untuk *supervisory control* berbasis *open source* dengan *pilot plant biodisel* sebagai contoh aplikasinya, merupakan salah satu wujud nyata untuk memulai solusi tersebut.

Java menjadi salah satu *tools* yang mampu menunjang upaya mengembangkan perangkat lunak yang memiliki kemudahan untuk dimodifikasi sesuai kebutuhan. Perangkat lunak juga dilengkapi dengan koleksi data HMI dengan format XML yang memungkinkan sistem lain dapat menggunakannya (*data interchange*). Protokol TCP/IP ditambahkan untuk membentuk komunikasi dengan RTU.

Dari sisi RTU, meski belum sepenuhnya mandiri, akan tetapi ketergantungan dapat dikurangi dengan menggunakan *controller* yang bersifat *universal*, di mana kita dapat memodifikasi ulang program di dalamnya sesuai kebutuhan dan kesepakatan dengan unit HMI.

2. Tinjauan pustaka

2.1 *Human machine interface*

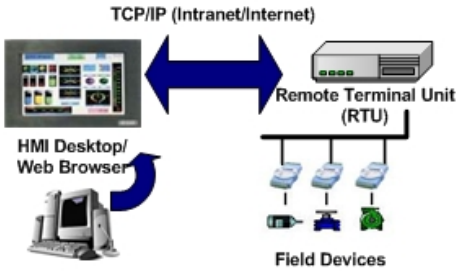
Seperti telah diurakan pada bagin pendahuluan bahwa HMI merupakan perangkat lunak antar muka berupa *Graphical User Interface* berbasis komputer yang menjadi penghubung antara operator dengan mesin atau peralatan yang dikendalikan serta bertindak pada level *supervisory* [1]. Beberapa referensi tentang HMI [1;2] menyebutkan bahwa secara umum HMI memiliki fungsi-fungsi seperti berikut:

- *Setting*
  Mengubah nilai ambang dari suatu paramater input (sensor) atau menentukan kondisi output (aktuator) berdasarkan nilai input yang diperoleh.
- *Monitoring*
  Mengawasi kondisi plant secara *real time*. Tampilan kondisi *plant* adalah berdasarkan hasil pembacaan input dan output dari proses yang sedang berlangsung pada *plant*.
- *Take action*
  Menjalankan dan memberhentikan suatu proses pada *plant*
- *Data Logging & Storage*
  Pengambilan dan penyimpanan data dalam suatu koleksi data. Pada umumnya data dapat berupa data pengukuran, status sistem yang diwakili oleh status valve-valve sebagai aktuator, status alarm,

tanggal pengambilan & penyimpanan data.

- *Alarm history* dan *summary*
  Menyimpan kondisi alarm, sehingga dapat diketahui alasan terjadinya penyimpangan dalam sistem
- *Trending*
  Suatu istilah untuk penampilan grafik dari sebuah proses., misalnya grafik penampil proses kenaikan dan penurunan suhu sistem. Trending dapat dilihat secara *online real time* atau historis.

Contoh visualisasi sederhana tentang bentuk interaksi HMI dengan RTU yang mengontrol sejumlah *filed device*, diperlihatkan pada gambar 1.



Gambar 1 Interaksi HMI dan RTU

HMI dan RTU saling berkomuniksi dalam dua arah. RTU mengendalikan jalannya proses sistem atau *plant* dengan memantau kondisi input, mengendalikan output, dan menerima permintaan data dari HMI untuk direspon dengan mengirim balik data yang diperlukan ke HMI. Selain mengirim permintaan data ke RTU, HMI juga mengkonfigurasi input output *plant* yang akan dipantau, mengatur jadwal *data logging and storage*, mengeksekusi, mengendalikan dan menghentikan proses pada *plant* (*take action*) sewaktu-waktu jika diperlukan.

Penggunaan protokol komunikasi TCP/IP di antara keduanya, memudahkan proses pada *plant* dapat diakses dan ditampilkan dalam *desktop browser* atau *web browser*, melalui intranet atau internet.

Fungsi *data logging and storage* memungkinkan HMI memiliki suatu koleksi data berupa sekumpulan file dengan format tertentu, seperti CSV atau format file teks lainnya. Setiap file dapat dibedakan berdasarkan waktu *logging*.

Ada dua perlakuan diberikan terhadap sekumpulan file yang diperoleh dari proses *data logging*, yaitu ditampilkan secara *online real time* dan disimpan untuk keperluan *Historical Trending*. Kemudian sesuai jadwal yang ditetapkan, baik secara manual atau otomatis, dilakukan konversi format koleksi data HMI ke format DBMS tertentu.

Konversi ke format DBMS umumnya diperlukan untuk keperluan analisis data lebih lanjut yang membutuhkan integritas data dengan sistem lain, seperti dalam sistem PAC [7]. Dalam sistem PAC, HMI adalah salah satu dari sejumlah sub sistem yang dapat saling bertukar data menggunakan protokol standar, melalui PAC sebagai unit pemroses utama.

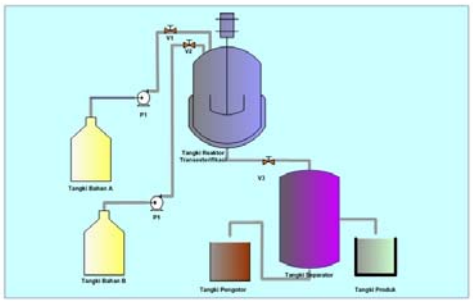
Teknologi XML dan beberapa teknologi internet lainnya seperti SOAP, WML dan XSL [6] telah digunakan untuk mengimplementasikan produk-produk *web-based* HMI [4;5]. Teknologi-teknologi tersebut menjadi dasar untuk pertukaran data, manipulasi data dan transmisi data.

Telah banyak sistem dibangun dengan menggunakan berbagai *tools* yang telah memiliki parser XML. Perihal mana yang terbaik untuk digunakan, apakah XML atau DBMS, tentunya itu tergantung kepada sistem yang membutuhkannya. Paramater *response time* yang sedikit dan *throughput* penyimpanan data yang besar membuat performansi DBMS lebih baik dari pada XML. Oleh karena itu jika kompleksitas sistem sangat tinggi, maka DBMS lebih dianjurkan demi mencapai ketepatan dan kecepatan penyampaian informasi [8]. Jika kompleksitas sistem rendah atau sedang, sistem dapat menggunakan data XML.

2.2 *Pilot plant bio-disel*

Plant yang dikendalikan adalah *pilot plant* untuk menghasilkan *biodiesel* (*metil ester*) dengan sistem kontinyu. Bahan dasar yang digunakan adalah minyak goreng dan metanol dengan tambahan katalis KOH. Ada beberapa proses yang terjadi dalam

pembuatan *biodiesel*, yaitu proses transesterifikasi yaitu proses bereaksinya bahan-bahan dalam tangki reaktor, proses pemisahan produk dari kotoran, proses pencucian, dan proses pemurnian. Batasan dari sistem yang akan digunakan adalah sampai pada tahap pemisahan produk dari pengotor. Visualisasi *pilot plant* dapat dilihat pada gambar 2.



Gambar 2 Pilot Plant Biodiesel

Pembentukan produk *metil ester* dengan sistem kontinyu melalui dua tahap utama, yaitu:

- tahap *Start Up*
- tahap kontinyu

Pada tahap *Start Up*, proses berlangsung dengan sistem *batch*, yaitu bahan-bahan dicampur dengan perbandingan tertentu, dimasukkan ke dalam tangki reaktor transesterifikasi, kemudian dilakukan pemanasan dan pengadukan selama waktu tertentu.

Setelah tahap *Start Up* tercapai, dimulailah tahap Kontinyu. Minyak goreng dari tangki bahan A dan campuran methanol dengan KOH dari tangki bahan B dialirkan menuju reaktor transesterifikasi secara bersamaan dengan perbandingan tertentu. Proses pengaliran dan pencampuran bahan A dan bahan B ini berlangsung secara terus menerus. Massa total semua bahan yang masuk ke dalam tangki reaktor harus sama dengan massa total semua bahan yang keluar dari tangki reaktor. Untuk mencapai total massa yang sama, kecepatan masing-masing bahan diatur mengikuti perbandingan tertentu. Pross pada sistem kontinyu pun berlangsung pada suhu dan kecepatan pengadukan tertentu.

Keluaran dari tangki reaktor yang merupakan hasil dari proses kontinyu, dialirkan menuju tangki separator. Keluaran ini pada tangki separator akan membentuk dua lapisan, yaitu produk *metil ester* pada lapisan atas, dan produk pengotor (*gliserol* dan *metanol*) pada lapisan bawah. Kedua produk ini dipisahkan dengan mengguanakn corong pemisah.

3. Metodologi

Metodologi yang digunakan dalam mengembangkan sistem adalah metodologi berorientasi obyek dengan menggunakan notasi UML sebagai *tools* untuk melakukan analisis dan perancangan sistem.

Analisis adalah proses menggali kebutuhan sistem dan menentukan apakah yang harus dilakukan oleh sistem untuk memenuhi kebutuhan tersebut. Tahapan analisis terdiri atas mengidentifikasi aktor, membuat model proses bisnis (*a simple bussiness process*) menggunakan diagram aktivitas UML, mengidentifikasi dan mengembangkan *use case* serta membuat diagram interaksi.

Aktor didefinisikan sebagai faktor atau obyek di luar sistem yang berinteraksi dengan sistem. Diagram aktivitas menggambarkan urutan aktivitas yang terjadi dalam sistem secara umum. Diagram *use case* menggambarkan urutan interaksi antara aktor dengan sistem untuk menjalankan suatu fungsi. Diagram interaksi memodelkan bentuk-bentuk interaksi yang terjadi di antara obyek-obyek yang terlibat dalam sistem. Jenis diagram interaksi yang digunakan untuk menganalisis kebutuhan perangkat lunak HMI ini adalah diagram sekuen.

Beberapa operator dalam diagram sekuen seperti *alt*, *ref* dan *loop* [9] digunakan untuk menganalisis kebutuhan perangkat lunak HMI. Operator *alt* digunakan untuk menunjukkan adanya kondisi pilihan (*if-else conditon*). Operator *ref* menginformasikan bahwa ada sejumlah interaksi dalam suatu diagram sequen yang merujuk kepada diagram sequen yang telah ada. Operator *loop* menunjukkan bahwa dalam diagram

sekuen tersebut terdapat sejumlah interaksi yang berulang.

Perancangan kelas-kelas dilakukan berdasarkan obyek-obyek yang teridentifikasi melalui analisis diagram sekuen. Kelas merupakan pola yang menggambarkan kumpulan obyek dengan kepemilikan sifat atau perilaku yang sama.

Sejumlah kelas hasil analisis dikelompokkan sesuai peranannya sebagai *boundary classes*, *entity classes* dan *control classes* [9]. *Boundary classes* merupakan kelas-kelas yang berhubungan langsung dengan lingkungan di luar sistem, seperti pengguna atau sistem lain. *Boundary classes* terbagi atas dua jenis kelas yaitu *user interface classes* dan *devices/system interface classes*. *Entity class* merupakan kelas yang menangani penyimpanan dan pengaksesan informasi entitas (*entity*) dalam sistem. *Control class* merupakan kelas yang menghubungkan *boundary class* dengan kelas pembentuk suatu fungsi yang sesuai dengan permintaan *boundary class*. Selain itu melalui peranan sebuah *control class*, e*ntity class* dapat menampilkan informasi yang dimilikinya dengan menggunakan sebuah *boundary class*.

Tahap terakhir adalah implementasi hasil perancangan ke dalam bentuk kode-kode, dengan menggunakan Java sebagai salah satu bahasa pemrograman berorientasi obyek.

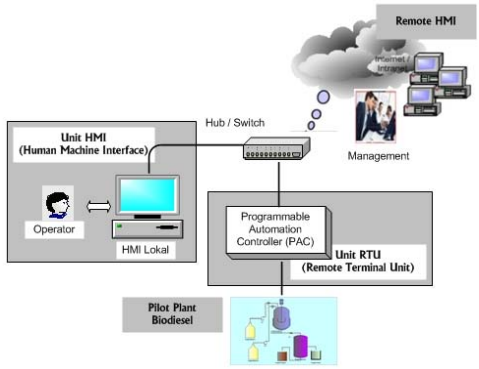
4. Desain dan pembahasan hasil

4.1 Gambaran umum sistem

Secara keseluruhan, sistem yang dikembangkan adalah *supervisory control* untuk pemantauan dan pengendalian terhadap *pilot plant biodiesel*. Gambar 3 memperlihatkan blok diagram sistem keseluruhan tersebut.

Sistem terdiri dari *pilot plant* yang didalamnya terpasang sensor untuk diamati nilainya dan aktuator dikendalikan kondisinya. Sensor dan aktuator terhubung ke unit RTU yang merupakan pengendali utama sistem pada *plant*. RTU juga memiliki unit akuisisi data hasil pengukuran.

Data hasil pengukuran dikirim ke unit HMI, berdasarkan permintaab dari HMI. Operator dapat memantau dan mengendalikan sistem *plant* melalui HMI. Komunikasi antara unit RTU dan HMI menggunakan komunikasi berbasis TCP/IP.



Gambar 3 Sistem yang dibangun

Bentuk pengendalian yang dilakukan HMI adalah berupa pengeksekusian dan penghentikan proses pada *plant*. Bentuk pengawasan HMI pada *plant* adalah dengan melakukan permintaan data ke RTU secara periodik untuk mendapatkan data hasil pembacaan sensor dan aktuator, sehingga kondisi sistem selalu terpantau.

Bentuk komunikasi lainnya adalah ketika HMI melakukan *upload* program ke RTU. Nilai-nilai hasil *setting* timer baik untuk proses *start up* maupun proses kontinyu disisipkan terlebih dahulu ke dalam program yang akan di-*upload* ke RTU.

HMI juga dapat menampilkan proses pada *pilot plant* melalui jaringan internet.

4.2 Gambaran umum perangkat lunak

Perangkat lunak yang dikembangkan adalah perangkat lunak HMI yang diperlukan untuk mendukung otomatisasi pembuatan biodiesel, mengawasi dan mengendalikan sistem, agar proses berjalan sesuai kebutuhan.

Secara umum, perangkat lunak HMI untuk sistem otomasi pada pilot plant biodiesel ini memiliki fungsi-fungsi sebagai berikut:

a. Setting

Mempersiapkan hal-hal yang berkaitan dengan proses terbentuknya komunikasi dan seputar proses pada *plant*, seperti:

- *Setting* IP dan Port milik RTU
- *Setting timer* untuk proses *start up* dan proses kontinyu.
- Menulis program untuk dijalankan di RTU sambil menyisipkan nilai timer sebagai *delay* proses.

b. Komunikasi
HMI menjalin komunikasi dengan RTU melalui jaringan berbasis TCP/IP untuk melakukan:

- *Upload* program ke RTU
- Mengeksekusi program di RTU
- Mengendalikan aktuator-aktutor pada *plant*
- Menghentikan program di RTU
- Meminta data dari RTU secara periodik

c. Melakukan pengelolaan terhadap data pengukuran yang telah diamibil dari RTU, dengan cara:

- Data diterima, dipilah dan langsung ditampilkan dalam bentuk numerik atau grafik online.
- Data diterima, dipilah dan disimpan dalam suatu file dengan XML. Data tersimpan ini dapat kembali ditampilkan dalam bentuk tabel atau grafik.

Tiga fungsi utama tersebut dapat mewakili fungsi umum dari suatu perangkat lunak HMI, seperti *setting*, *monitoring*, *take action*, *data logging and storage*, *alarm history and summary* dan *trending*.

4.3 Analisis dan perancangan perangkat lunak

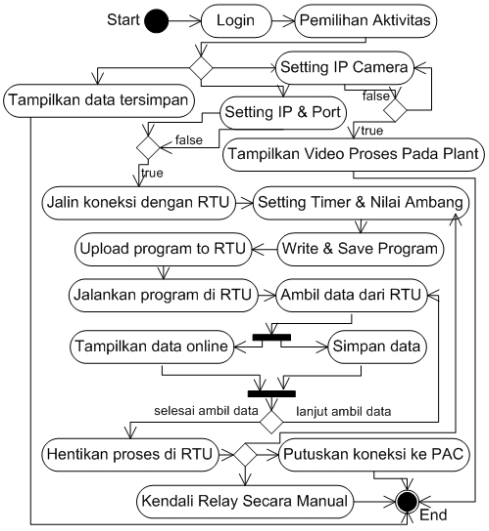
4.3.1 Identifikasi aktor

Dalam sistem HMI yang dikembangkan, aktor yang teridentifikasi ada dua, yaitu operator dan RTU. Operator berperan mengirim permintaan data kepada sistem atau RTU, sementara RTU memberikan data yang dibutuhkan kepada HMI.

4.3.2 Pengembangan diagram aktifitas

Diagram aktivitas dari perangkat lunak HMI ditampilkan pada gambar 4. Diagram ini memperlihatkan urutan aktivitas keseluruhan yang terjadi dalam proses pemantauan dan pengendalian terhadap *pilot plant biodiesel*.

Dalam kondisi tidak melakukan komunikasi dengan RTU, aktivitas yang dapat dilakukan adalah menampilkan data tersimpan dan menampilkan kondisi *plant* pada RTU melali *browser*.



Gambar 4 Diagram Aktivitas HMI

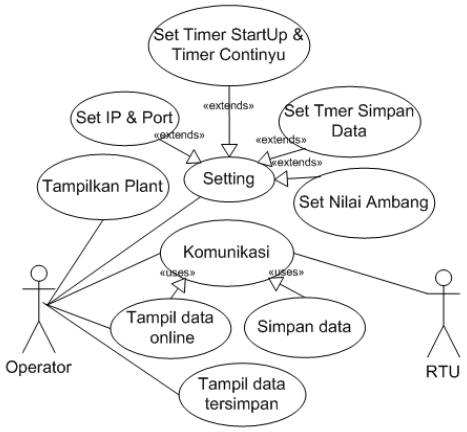
4.3.3 Pengembangan diagram *use case*

Berdasarkan fungsi-fungsi umum dari perangkat lunak yang telah dijelaskan dalam gambaran umum perangkat lunak, dibuatlah diagram *use case* seperti diperlihatkan oleh gambar 5.

Use case Setting mengelola empat jenis setting, yaitu *setting* IP dan port milik RTU, setting timer untuk proses *startup* dan kontinyu, setting timer untuk menyimpan data dari RTU dan setting nilai-nilai ambang pengukuran yang diperbolehkan.

Use case Komunikasi menangani proses koneksi di antara HMI dengan RTU, di mana proses membuka dan menutup komunikasi menjadi wewenang HMI melalui instruksi operator. *Use case* ini juga

menangani penerimaan data dari RTU dan penyimpanan data sementara di HMI.



Gambar 5 Diagram Use Case HMI

Data yang diterima dari RTU, disimpan sementara untuk keperluan penampilan data *online* oleh *use case* Tampil data *online* dan untuk keperluan penyimpanan data secara permanen oleh *use case* Simpan data yang berjenis *abstract use case*.

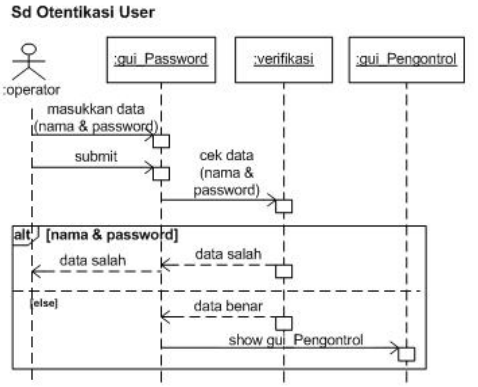
Use case penampil lainnya adalah *use case* yang menangani penampilan data tersimpan dan *use case* yang menampilkan kondisi *plant* pada RTU melalui *browser*.

4.3.4 Pengembangan diagram sekuen

Analisa perilaku sistem selanjutnya adalah mengidentifikasi obyek-obyek yang saling berinteraksi dalam sistem melalui penampilan sembilan diagram sekuen. Setiap diagram sekuen memodelkan interaksi obyek-obyek dalam menjalankan satu fungsi yang menunjang keseluruhan fungsi sistem. Kesembilan diagram sekuen tersebut, memodelkan fungsi-fungsi otentikasi pengguna, pembukaan koneksi dengan RTU, pengeksekusian program di RTU, pengambilan data RTU, penampilan data RTU, penyimpanan data RTU, pengendalian relay, parsing FileXMLTerseleksi, dan penampilan data terseleksi.

Diagram sekuen pertama diperlihatkan oleh gambar 6.

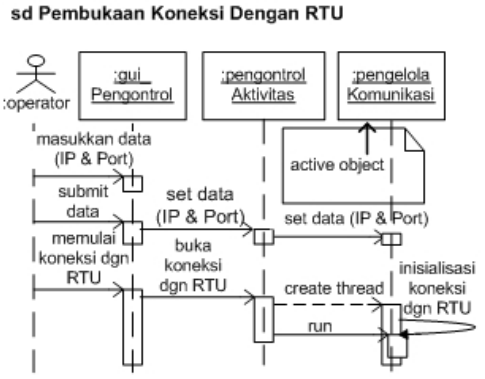
Diagram sekuen pertama memodelkan interaksi obyek-obyek yang terlibat dalam fungsi ontentikasi pengguna. Urutan interaksi yang terjadi adalah memasukkan nama pengguna, memasukkan *password* dan melakukan verifikasi nama dan *password* untuk masuk ke sistem HMI.



Gambar 6 Diagram Sekuen Otentikasi Pengguna

Obyek verifikasi melakukan pengecekan atas nama dan *password* yang dimasukkan oleh operator. Obyek gui_password menjadi antar muka antara operator dengan obyek verifikasi. Obyek gui_pengontrol merupakan obyek panel utama yang mengeloa aktivitas obyek-obyek panel lainnya.

Diagram sequen kedua memodelkan interaksi obyek-obyek yang terlibat dalam fungsi pembukaan koneksi dengan RTU, seperti tampak pada gambar 7. Urutan interaksi yang terjadi adalah *setting* nomor IP dan nomor *port* serta membuka koneksi dengan RTU.

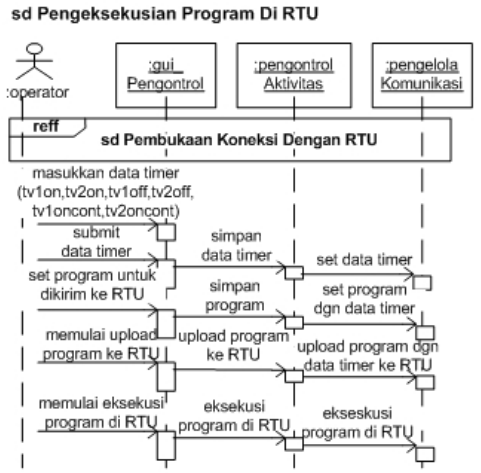


Gambar 7 Diagram Sekuen Pembukaan Koneksi Dengan RTU

Tahapan komunikasi dengan RTUmerupakan suatu *thread* tersendiri yang diciptakan oleh obyek pengelolaAktivitas

dan diskesekusi olehnya melalui metode *run* yang dimiliki obyek pengelolaKomunikasi. Eksekusi *thread* diakhiri oleh kondisi yang menyebabkan terputusnya komunikasi dgn RTU.

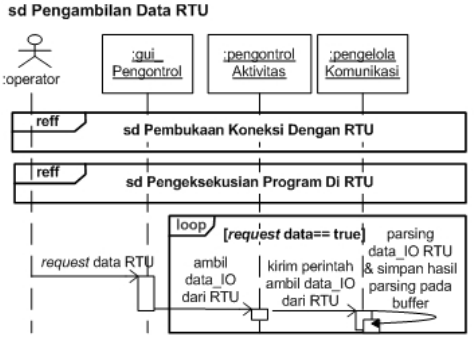
Diagram sekuen ketiga memodelkan interaksi obyek-obyek yang terlibat dalam fungsi pengeksekusian program di RTU. Diagram sekuen ini dapat dilihat pada gambar 8.



Gambar 8 Diagram Sekuen Pengeksekusian Program Di RTU

Urutan interaksinya adalah membuka koneksi dengan RTU, *seting* timer untuk proses *startup* dan kontinyu, *setting* program untuk ke RTU, kirim (*upload*) program ke RTU dan jalankan program di RTU untuk memulai proses pada *plant*.

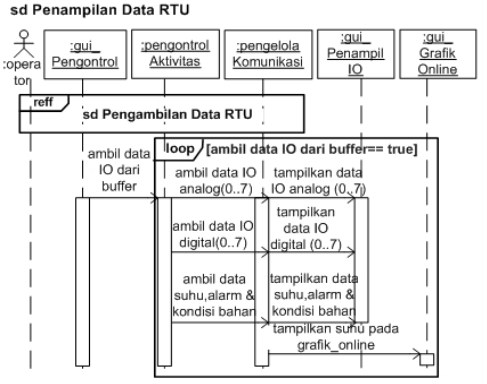
Diagram sekuen keempat memodelkan interaksi obyek-obyek yang terlibat dalam fungsi pengambilan data RTU, seperti terlihat pada gambar 9.



Gambar 9 Diagram Sekuen Pengambilan Data RTU

Urutan interaksinya adalah membuka koneksi dengan RTU, mengeksekusi program di RTU, meminta data ke RTU, menerima data dari RTU, menguraikan data RTU (*parsing data*) menjadi data-data suhu, alarm, kondisi bahan dan data-data input output lainnya, kemudian menyimpan sementara data-data hasil penguraian tersebut.

Diagram sekuen kelima memodelkan interaksi obyek-obyek yang terlibat dalam fungsi penampilan data RTU secara *online*.



Gambar 10 Diagram Sekuen Penampilan Data RTU

Seperti diperlihatkan pada gambar 10, maka urutan interaksi dalam proses penampilan data RTU adalah:

Mengambil data-data suhu, alarm, kondisi bahan dan data input output lainnya, dari penyimpanan sementara.

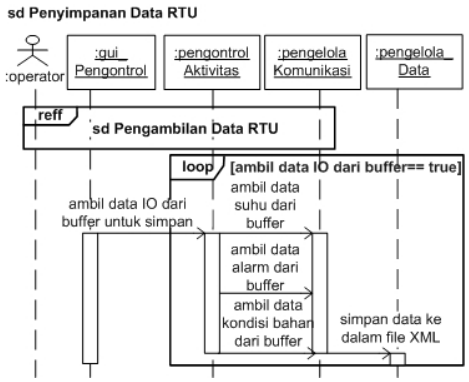
Menampilkan data-data tersebut melalui obyek gui_PenampilIO.

Menampilkan data suhu dalam bentuk grafik melalui obyek guiGrafikOnline.

Diagram sekuen keenam memodelkan interaksi obyek-obyek yang terlibat dalam fungsi penyimpanan data dari RTU, yang diperlihatkan oleh gambar 11.

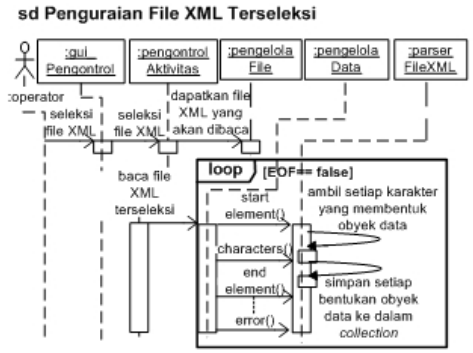
Urutan interaksi dalam proses penyimpanan data secara permanen adalah mengambil data-data suhu, alarm dan kondisi bahan dari penyimpanan sementara, mengemas data-data tersebut dalam bentuk obyek-obyek data (obyek datapengukuran),

kemudian menyimpan obyek-obyek data tersebut ke dalam format XML.



Gambar 11 Diagram Sekuen Penyimpanan Data RTU

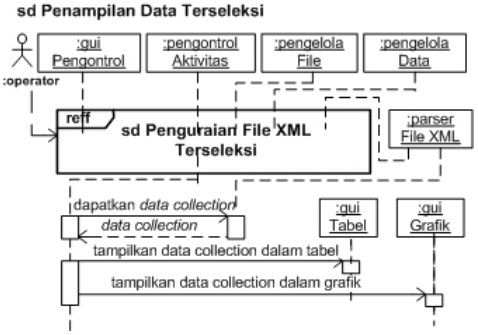
Diagram sekuen ketujuh memodelkan interaksi obyek-obyek yang terlibat dalam fungsi penguraian file XML terseleksi, yang ditampilkan pada gambar 12.



Gambar 12 Diagram Sekuen Penguraian File XML Terseleksi

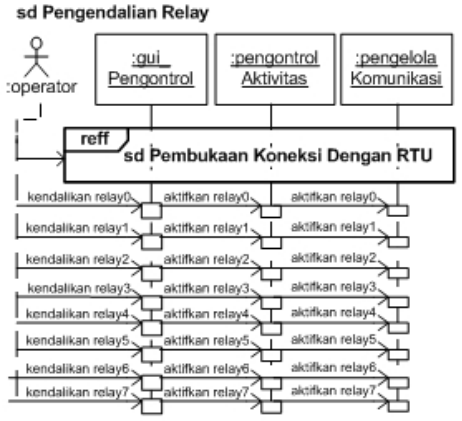
Urutan interaksi dalam proses penguraian file XML terseleksi adalah menyeleksi file XML yang akan dibaca, membaca dan menguraikan elemen-elemen data berformat XML menjadi data-data yang diperlukan, mengemas data-data hasil penguraian tersebut menjadi sebuah obyek data pengukuran, kemudian menyimpan setiap obyek data pengukuran yang terbentuk ke dalam suatu *collection*. Penyimpanan ke dalam suatu *collection*, untuk setiap obyek data pengukuran yang terbentuk dari hasil penguraian elemen-elemen data XML, akan terus berulang hingga penelusuran isi file XML berakhir.

Diagram sekuen kedelapan yang tampak pada gambar 13, memodelkan interaksi obyek-obyek yang terlibat dalam proses penampilan data terseleksi. Proses penampilan data terseleksi ini melibatkan penyeleksian dan penguraian file XML yang menghasilkan sejumlah obyek data pengukuran dalam *collection*. Urutan interaksi selanjutnya adalah mendapatkan atau membaca sejumlah obyek data pengukuran dari *collection*, kemudian menampilkannya dalam bentuk tabel dan grafik, melalui obyek guiTabel dan guiGrafik.



Gambar 13 Diagram Sekuen Penampilan Data Terseleksi

Diagram sekuen kesembilan yang tampak pada gambar 14, memodelkan interaksi obyek-obyek yang terlibat dalam fungsi pengendalian relay-relay pada *plant*.

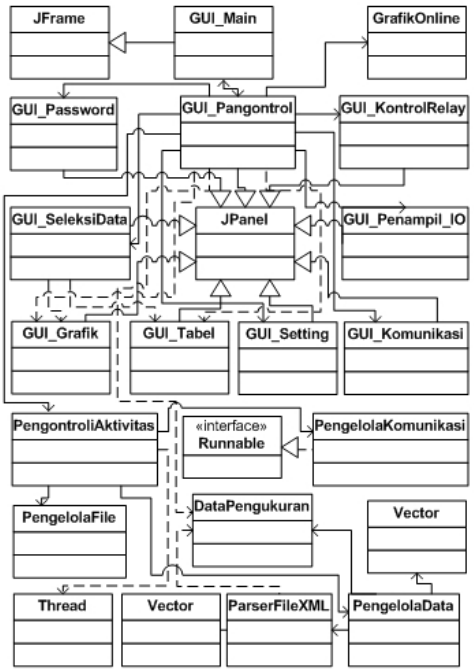


Gambar 14 Diagram Sekuen Pengendalian Relay

Saat akan melakukan pengendalian terhadap relay-relay pada *plant* untuk kebutuhan tertentu, harus dipastikan bahwa koneksi dengan RTU masih terbentuk dan proses pada *plant* yang bersifat otomatis dihentikan terlebih dahulu melalui perintah dari operator HMI untuk menghentikan program di RTU.

4.3.5 Pengembangan diagram kelas

Obyek-obyek teridentifikasi yang dihasilkan pada tahap pengembangan diagram sekuen menjadi dasar untuk perancangan kelas-kelas. Diagram kelas menggambarkan struktur dan deskripsi kelas serta hubungan antara satu kelas dengan kelas lainnya. Diagram kelas dari HMI secara keseluruhan dapat dilihat pada gambar 15.



Gambar 15 Diagram Kelas HMI

Kelompok *boundary classes* berjenis *user interface classes* pada diagram kelas HMI ini memiliki tiga fungsi, yaitu sebagai:

Panel penampil data, yang terdiri dari kelas GUI_Penampil_IO, kelas GUI_Tabel, kelas GUI_Grafik, dan kelas GrafikOnline.

Panel penginput data, yang terdiri dari kelas GUI_Password, kelas GUI_KontrolRelay, kelas GUI_SeleksiData, kelas GUI_Setting, dan GUI_Komunikasi.

Komunikator di antara sesama kelas panel dalam kelompok *boundary classes*, yaitu kelas GUI_Pengontrol.

Kelas GUI_Pengontrol juga menjadi kelas penghubung antara kelompok *boundary classes* dengan kelompok *control classes* atau dengan kelompok *entity classes*. Pemilihan l*ayout manager* berjenis *card layout* pada tahap implementasi, menjadikan kelas GUI_Pengontrol sebagai panel dasar untuk menempatkan sejumlah panel-panel yang ada. Dengan konsep *card layout* memungkinkan sejumlah panel dapat ditampilkan dalam satu panel yang sama secara bergantian.

Kelas PengelolaKomunikasi termasuk dalam kelompok *boundary classes* berjenis *system/device interface classes* pada diagram kelas HMI. Kelas PengelolaKomunikasi berinteraksi dengan RTU melalui pengiriman *request command* ke RTU dan penerimaan data dari RTU.

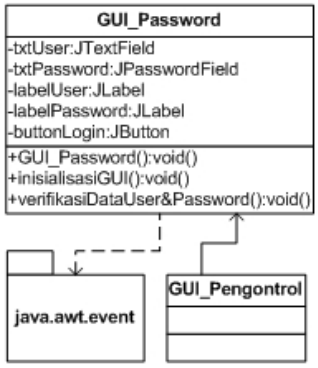
Kelas PengontrolAktivitas merupakan *control class* yang mengatur kebutuhan interaksi antara kelas GUI_Pengontrol dengan kelas-kelas lainnya untuk membentuk fungsi tertentu. Kelas DataPengukuran merupakan *entity class* yang berinteraksi dengn sistem penyimpanan data untuk menyimpan dan menampilkan informasi data pengukuran.

Untuk mengetahui bentuk hubungan antara satu kelas dengan kelas lainnya, maka penulis mengelompokkan kelas-kelas tersebut dalam delapan kelompok diagram kelas, sesuai dengan fungsi masing-masing.

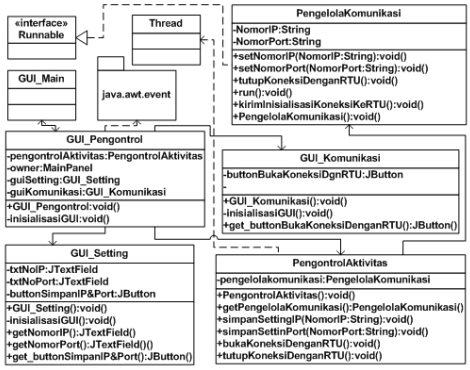
Diagram kelas pertama memperlihatkan hubungan antar kelas yang berkaitan dengan fungsi otentikasi pengguna, seperti tampak pada gambar 16.

Pada dasarnya kelas GUI_Password merupakan atribut dari kelas GUI_Pengontrol sebagai *boundary class*, di mana nama dan *password* yang bernilai *valid* akan mengaktifkan obyek panel utama yang merupakan jelmaan dari kelas GUI_Pengontrol.

Diagram kelas kedua memperlihatkan bentuk hubungan antar kelas-kelas yang membentuk fungsi pembukaan koneksi dengan RTU, seperti pada gambar 17.



Gambar 16 Diagram Kelas Otentikasi Pengguna



Gambar 17 Diagram Kelas Pembukaan Koneksi Dengan RTU

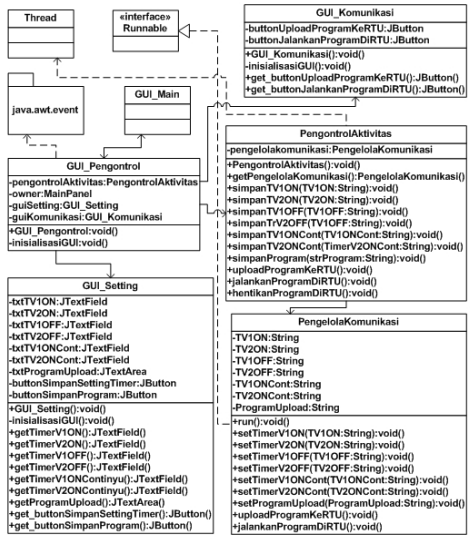
Diagram kelas ketiga memperlihatkan bentuk hubungan antar kelas-kelas yang membentuk fungsi pengeksekusian program di RTU, seperti pada gambar 18.

Gambar 19 memperlihatkan diagram kelas yang menjelaskan hubungan antar kelas-kelas dalam menjalankan fungsi pengambilan data RTU.

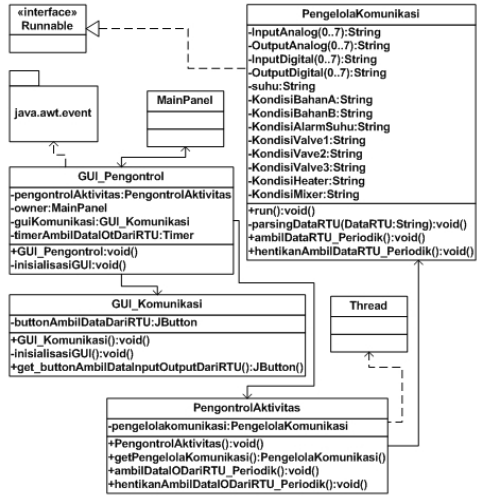
Diagram kelas kelima memperlihatkan bentuk hubungan antar kelas-kelas yang membentuk fungsi penampilan data RTU, seperti pada gambar 20.

Diagram kelas keenam memperlihatkan hubungan antar kelas-kelas yang membentuk fungsi penyimpanan data RTU, seperti pada gambar 21.

Gambar 22 memperlihatkan bentuk hubungan antar kelas-kelas yang menjalankan fungsi pengendalian relay-relay pada *plant*.



Gambar 18 Diagram Kelas Pengeksekusian Program Di RTU

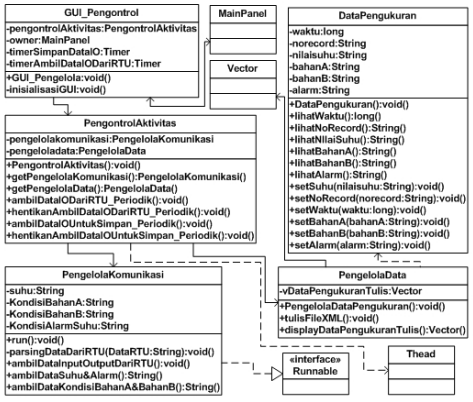


Gambar 19 Diagram Kelas Pengambilan Data RTU

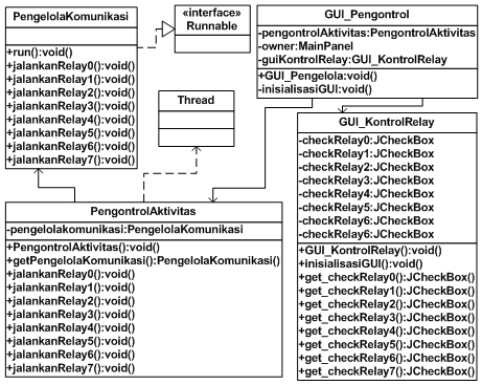
Dari keenam diagram kelas, yang telah ditampilkan, selain diagram kelas otentikasi pengguna,, tampak bahwa kelas PengelolaKomunikasi memiliki peranan dalam memulai komunikasi dengan RTU,

mengeksekusi program di RTU, mengambil data RTU, menampilkan data RTU, menyimpan data RTU, dan mengendalikan relay-relay pada *plant*.

Gambar 20 Diagram Kelas Penampilan Data RTU



Gambar 21 Diagram Kelas Penyimpanan Data RTU



Gambar 22 Diagram Kelas Pengendalian Relay

Kelas PengelolaKomunikasi mengimplementasikan metode *run*() milik *interface Runnable,* untuk memulai tahapan-tahapan dalam menjalin koneksi di antara TCP *socket* pada sisi HMI sebagai *client* dan TCP *socket* pada sisi RTU sebagai *server*. Metode *run*() tersebut diaktifkan oleh kelas PengelolaKomunikasi dengan memanfaatkan sebuah kelas *Thread*. Ketika operator memulai untuk membuka koneksi dengan RTU, maka secara otomatis terjadi pengaktifan metode run(). Cuplikan dari bentuk implementasi pengaktifan metode run() dalam kode-kode java adalah *new Thread(new PengelolaKomunikasi).start()*

Kelas PengelolaAktivitas mengatur berjalannya fungsi-fungsi HMI secara keseluruhan. Untuk fungsi-fungsi seperti pembukaan koneksi dengan RTU, pengeksekusian program di RTU dan pengambilan data RTU, kelas PengelolaAktivitas berinteraksi dengan kelas PengelolaKomunikasi untuk berkomunikasi dengan RTU dan Kelas GUI_Pengontrol sebagai antar muka bagi operator untuk mengeksekusi setiap fungsi-fungsi tersebut.

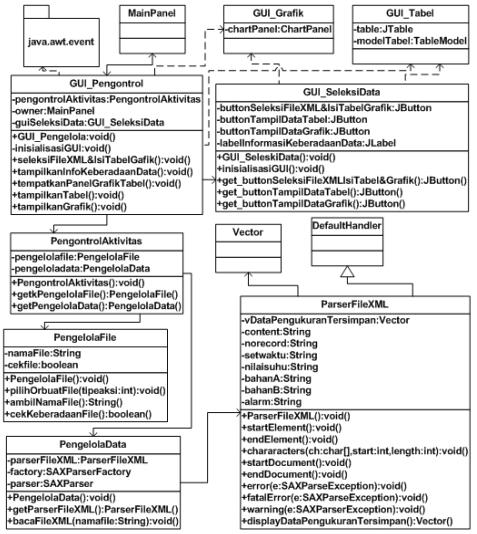
Untuk menjalankan fungsi penampilan data RTU, kelas PengelolaKomunikasi berinteraksi dengan kelas PengelolaKomunikasi dan kelas GUI_PenampilInputOutput serta kelas GUI_GrafikOnline sebagai antar muka bagi operator untuk memantau kondisi input output pada *plant*.

Kelas PengelolaAktivitas berinteraksi dengan kelas PengelolaKomunikasi, kelas PengelolaData, dan kelas DataPengukuran untuk menjalankan fungsi penyimpanan data dalam format XML serta kelas GUI_Pengontrol sebagai antar muka bagi operator untuk mengeksekusi fungsi penyimpanan data XML tersebut.

Untuk menjalankan fungsi penampilan data terseleksi, kelas PengelolaAktivitas beriteraksi dengan kelas PengelolaFile untuk penyeleksian file XML yang akan dibaca, kemudian berinteraksi dengan kelas PengelolaData untuk pembacaan data dari

file XML terseleksi tersebut. Setelah itu menampilkan data file XML terseleksi ke dalam bentuk grafik dan tabel yang melibatkan kelas GUI_Grafik dan kelas GUI_Tabel sebagai antar muka bagi operator untuk melihat data hasil penyeleksian. Kelas GUI_Pengontrol tetap berlaku sebagai antar muka bagi operator untuk mengeksekusi penyeleksian file XML.

Gambar 23 memperlihatkan interaksi antar kelas-kelas yang menjalankan fungsi penampilan data terseleksi.



Gambar 23 Diagram Kelas Penampilan Data Terseleksi

4.3.6 Perancangan struktur data XML

Struktur data XML mengikuti aturan berikut ini:

- Standar Header
  ```
  <?xml version=”1.0”?>
  <!DOCTYPE RawData SYSTEM
  ”101109.dtd”>
  ```
  Dua baris deklarasi ini ditulis di awal dokumen dimulai dari baris paling atas, yang menunjukkan versi XML , nama file berekstensi dtd yang berisi referensi tentang nama-nama elemen yang tertulis dalam struktur data XML, dan nama root elemen dalam struktur tersebut
- Penulisan element root dan sejumlah element child yang terkandung di dalamnya

Berikut ini adalah bentuk struktur data file XML untuk perangkat lunak HMI.

```
<?xml version="1.0"?>
<!DOCTYPE RawData SYSTEM
"101109.dtd">
<RawData>
<DataPengukuran>
<NoRecord>data nomor
record</NoRecord>
<Waktu>data waktu</Waktu>
<NilaiSuhu>data suhu</NilaiSuhu>
<LevelInd1>data level
Ind1</LevelInd1>
<LevelInd2>data level
Ind2</LevelInd2>
<Alarm>data alarm</Alarm>
</DataPengukuran>
</RawData>
```

Elemen `<DataPengukuran>` beserta anak-anak elemennya akan berulang sebanyak jumlah obyek data pengukuran yang diperoleh dari RTU untuk setiap periode penerimaan. LevelInd1 dan LevelInd2 adalah elemen-elemen untuk untuk menampung data level indikator1 dan data level indikator2.

Isi dari file dtd sebagai referensi yang menjelaskan setiap elemen dalam struktur data XML adalah seperti berikut.

```
<?xml version="1.0"
encoding="UTF-8"?>
<!ELEMENT RawData
(DataPengukuran*)>
<!ELEMENT DataPengukuran
(NoRecord,Waktu,NilaiSuhu,LevelIn
d1,LevelInd2,Alarm)>
<!ELEMENT NoRecord (#PCDATA)>
<!ELEMENT Waktu (#PCDATA)>
<!ELEMENT NilaiSuhu (#PCDATA)>
<!ELEMENT LevelInd1 (#PCDATA)>
<!ELEMENT LevelInd2 (#PCDATA)>
<!ELEMENT Alarm (#PCDATA)
```

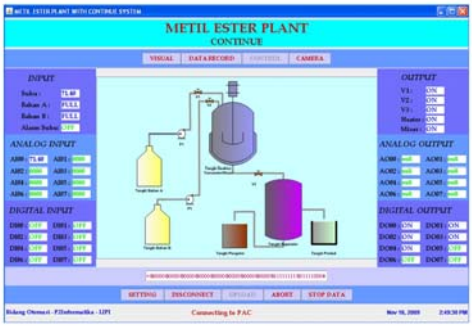
4.4 Implementasi

Bentuk implementasi dari hasil perancangan adalah kode-kode dalam bahasa java yang ketika dijalankan akan tampil seperti pada beberapa gambar berikut. Gambar 24 memperlihatkan menu utama dalam kondisi sedang menampilkan data RTU secara *online* pada panel penampil input dan output.

Gambar 25 memperlihatkan menu utama dalam kondisi sedang menampilkan data RTU secara *online* dalam bentuk grafik.

Gambar 26 memperlihatkan panel penampil *setting* IP dan Port RTU serta *timer* proses *start up* dan kontinyu.

Gambar 27 memperlihatkan panel penampil data XML terseleksi dalam bentuk tabel.



Gambar 24 Menu Utama Tampil Online

Gambar 25 Menu Utama Grafik Online

Gambar 26 Menu Utama Setting Timer

Gambar 27 Menu Utama Grafik Record Data

Gambar 28 memperlihatkan panel penampil data XML terseleksi dalam bentuk grafik.

Gambar 28 Menu Utama Tabel Record Data

5. Kesimpulan

Perangkat lunak *Human Machine Interface* untuk *supervisory control* pada *pilot plant biodiesel* berbasis *open source* yang dikembangkan, meskipun belum memiliki kompleksitas setinggi produk-produk berlisensi, akan tetapi secara umum dapat memenuhi fungsi-fungsi standar sebagai perangkat lunak HMI, seperti *setting*, *monitoring*, *take action*, *data logging and storage*, *alarm history* and *summary* dan *trending*.

Penggunaan protokol standar TCP/IP dan teknologi XML, mempersiapkan perangkat lunak HMI ini untuk mendukung konsep *data interchange* dalam suatu integritas sistem yang saling membutuhkan. Kemandirian dalam mengembangkan perangkat lunak, harus dapat dimanfaatkan untuk mengembangkan perangkat lunak HMI ke arah yang lebih universal, sehingga

tidak hanya tergantung pada satu produk RTU dengan format data tertentu.

6. Daftar pustaka

[1] GlobalSpec Industrial Controls Software, *Learn More About Human Machine Interface Software(HMI)*, diakses dari http://www.globalspec.com/LearnMore/Industrial_Engineering_Software/Industrial_Controls_Software/Human_Machine_Interface_Software_HMI, pada tanggal 6 Januari 2010.

[2] Juni Ardi Irawan, *HMI or MMI*, diakses dari http://juare97.wordpress.com, pada tanggal 18 Agustus 2009.

[3] Handi Wicaksono, *Introduction to SCADA*, diakses dari http://learnautomation.wordpress.com, pada tanggal 14 Mei 2009.

[4] Movicon, *SCADA/HMI XML based*, diakses dari http://www.automationwarehouse.com.au/movicon/x2.asp, pada tanggal 6 Januari 2010.

[5] PaperSearchEngine, *CT HMI – Web Enabled HMI Sofware*, diakses dari http://www.ctc-control.com/customer/techinfo/data/ct_hmi.pdf, pada tanggal 8 Januari 2009.

[6] InduSoft, *Using Internet Technologies To Create Web-Based HMI*diakses dari http://www.indusoft.com/PDF/Web-based_HMI.PDF, pada tanggal 8 Januari 2010

[7] Opto 22, *Understanding Programmable Automation Controllers (PACs)*", diakses dari http://www.opto22.com/documents/1634_PAC_White_Paper.pdf, pada tanggal 10 Januari 2010.

[8] Ragil Martha Ardianto, *Analisa Performansi XML dan DBMS sebagai Penyimpanan Data*, diakses dari http://www.ittelkom.ac.id/library/index.php?option=com_repository&Itemid=34&task=detail&nim=113050064, pada tanggal 10 Januari 2010.

[9] Zvika Gutterman, Adam Carmi, *UML Class Diagram and Packages*, diakses dari http://webcourse.cs.technion.ac.il/234321/Spring2005/ho/WCFiles/07-class-diagrams.ppt, pada tanggal 10 Januari 2010.

[10] Fowler_ch04.fm, *Sequence Diagram*, diakses dari http://www.pearsonhighered.com/assets/hip/us/hip_us_pearsonhighered/samplechapter/0321193687.pdf, pada tanggal 15 Januari 2010.

[11] Jeff Valdez, *UML Interaction Diagrams*, diakses dari http://www.docstoc.com/docs/16827517/15-UML-INTERACTION-DIAGRAMS, pada tanggal 15 Januari 2010.

[12] Ali Bahrami, *Object Oriented Systems Development*, Irwin/McGraw-Hill, Singapore, 1999.

[13] Kenneth L Calvert, Michael J. Donahoo, *TCP/IP Socket in Java Practical Guide for Programmers*, Morgan Kaufmann Publisher, University of Kentucky, Lexington, KY, USA.

[14] Rika Sustika, Endang Suryawati, "Pengembangan HMI (Human Machine Interface) untuk Sistem Kontrol dan Monitoring Pilot Plant Metil Ester dengan Bahasa Pemrograman Java", *Prosiding Seminar Nasional OSS III*, Bandung, 2009.

[15] Rika Sustika, Endang Suryawati, Oka Mahendra, Djohar Syamsi, "Supervisory Control Berbasis TCP/IP untuk Otomasi Pilot Plant Sistem Kontinu", *Prosiding Seminar Nasional Riset Teknologi Informasi*, Yogyakarta, 2009.