PEMBANGUNAN PERANGKAT LUNAK BERBASIS KOMPONEN STUDI KASUS: SISTEM INFORMASI AKADEMIK TERDISTRIBUSI

Sandra Islama Putra
Laboratorium Rekayasa Perangkat Lunak dan Sistem Informasi
Jurusan Teknik Informatika, Fakultas Teknik, Universitas Pasundan Bandung
Jalan Setiabudi No 193 Bandung
E-mail: syss1998@plasa.com

Abstrak

Model komputasi Client/Server merupakan salah satu teknologi yang digunakan dalam bidang pengembangan perangkat lunak. Salah satu aspek yang menarik untuk dikaji dalam model ini adalah teknologi pengembangan perangkat lunak berbasis komponen. Makalah ini membahas mengenai teknologi pengembangan perangkat lunak berbasis komponen dan penerapannnya dalam suatu studi kasus.

***Kata kunci**: CBSD, Komponen,Aplikasi Terdistribusi*

1. Pendahuluan

Model komputasi Client/Server, selanjutnya disebut model komputasi terdistribusi tengah menjadi isu yang menarik dan hangat diperbincangkan dalam dunia teknologi informasi khususnya dalam bidang pengembangan perangkat lunak. Salah satu aspek yang menarik untuk dikaji dalam model ini adalah teknologi pengembangan perangkat lunak berbasis komponen.

Dikatakan menarik karena tujuan dari teknologi ini adalah untuk mengurangi waktu dan usaha yang diperlukan dalam pengembangan suatu perangkat lunak. Hal ini disebabkan karena suatu komponen dimungkinkan untuk melakukan komunikasi dengan komponen yang lainnya sehingga dapat digunakan oleh banyak aplikasi yang besar dan beragam dalam suatu jaringan komputer. Teknologi ini juga sering disebut dengan istilah *middleware*, maksudnya komponen yang dibuat dijadikan sebagai penghubung antara *front-end* dan *back-end application*.

Teknologi berbasis komponen dimplementasikan oleh Microsoft dengan teknologi COM dan DCOM (*Distributed Common Object Model*), OMG (Object Management Group) mengimplementasikannya dengan teknologi CORBA (*Common Object Request Broker Architecture*), sedangkan sun mengimplementasi-kannya dengan JavaBean dan Entreprise Java Bean (EJB). Ketiga teknologi ini dirancang dengan tujuan yang sama, yaitu sebagai *platform* independen yang memungkinkan terjadinya komunikasi antar objek dalam jaringan komputer.

Di dalam makalah ini, akan dilakukan kajian terhadap teknologi pengembangan perangkat lunak berbasis komponen sekaligus menerapkannya dengan cara membuat komponen-komponen terdistribusi (**DISTSIMAKServer**).

Untuk menguji komponen yang dibuat, maka dalam makalah ini komponen yang dibuat akan diimplementasikan dalam studi kasus protipe Sistem Informasi Akademik berbasis komponen terdistribusi (**DISTSIMAK**) di Jurusan Informatika UNPAS Bandung.

2. Komponen dan Aplikasi Terdistribusi

Beberapa tahun terakhir ini teknologi pengembangan perangkat lunak berbasis komponen telah menjadi sebuah paradigma baru dalam mengembangkan suatu aplikasi perangkat lunak terutama pada aplikasi terdistribusi. Keuntungan dari teknologi ini adalah untuk mengurangi waktu dan biaya pengembangan, serta meningkatkan derajat dari interoperabilitas, portabilitas dan maintainabilitas. **[CRN97].** Bagian ini akan memberikan gambaran tentang beberapa aspek dari teknologi komponen.

2.1 Definisi Komponen

Komponen adalah entitas yang dapat digunakan oleh beberapa program yang berbeda. Komponen menyediakan model standar untuk pemaketan layanan-layanan. Pada sebuah aplikasi yang menggunakan komponen, komponen hanyalah sebuah kotak hitam, karena semua data dan implementasi detail yang dimiliki oleh sebuah komponen disembunyikan. Layanan dari sebuah komponen dibuka melalui *public interfaces*. Saat ini *tools* untuk pembuatan komponen telah mendukung penggunaan *interface*. [BRI01]

Dari literatur yang lain, istilah komponen adalah bagian dari perangkat lunak dalam bentuk *binary form,* dapat disebarkan (*deployed*) secara bebas (dapat dimuat kedalam sistem secara dinamis, ataupun diganti secara dinamis). Komponen harus memiliki mekanisme yang memungkinkan untuk berintegrasi dengan sistem tanpa modifikasi dan mengembangkan ulang sistem. **[KIR98]**

Suatu komponen dianggap sebagai *language-neutral*. Artinya bahwa sejumlah komponen dapat ditulis dalam bahasa yang berbeda. Ini berarti bahwa

ISBN: 979-756-061-6

meskipun komponen-komponen tersebut dirancang dan disebarkan dalam bahasa yang berbeda namun mereka dapat bekerja bersama. **[BRO00]**

Suatu komponen dapat diimplementasikan terbebas dari komponen yang lainnya. Hal ini dimungkinkan sebab komponen terenkapsulasi, dimana masing-masing komponen memiliki unit kecil dari pengembangan dan pengujian. Suatu komponen tidak dibatasi untuk dapat dijalankan pada platform tunggal, karena dimungkinkan untuk membuat *deployment* yang berbeda untuk suatu komponen agar dapat dijalankan pada beberapa platform. **[BRI01]**

Meskipun sebagian besar komponen dibuat untuk memenuhi kebutuhan dari suatu aplikasi, namun setiap kali suatu komponen dibuat dan disebarkan, maka sangatlah mungkin untuk menggunakan komponen tersebut pada aplikasi yang berbeda. Selama *interface* dari komponen tersebut memenuhi kebutuhan pengguna, maka komponen yang sama dapat digunakan untuk mengembangkan atau meningkatkan aplikasi yang lain. **[KIR98]**

Kegunaan dari komponen adalah untuk menyediakan beberapa layanan yang dapat digunakan oleh sistem yang berada di lingkungannya. Lingkungan dari komponen dapat terdiri dari komponen lain, aplikasi, dan lain sebagainya. Layanan yang disediakan oleh komponen kepada lingkungannya diakses melalui satu atau lebih *interface*. Spesifikasi *Interface* dapat dipandang sebagai sebuah kontrak, yang terjadi antara komponen yang menyediakan (mengimplementasi) layanan tertentu dan lingkunan yang menggunakan layanan tersebut. **[BRO00]**

Komponen terdiri dari beberapa properti yang menggambarkan karakteristik dari komponen tersebut. Komponen seharusnya bersifat *composable,* artinya sebuah komponen bukanlah merupakan sebuah aplikasi lengkap, melainkan merupakan bagian dari sebuah keseluruhan sebuah aplikasi. Dengan kata lain, sebuah komponen seharusnya menyediakan beberapa fungsionalitas yang digabungkan dengan fungsionalitas yang disediakan oleh komponen lain untuk memperoleh aplikasi yang lengkap. Suatu komponen harus dapat digunakan oleh beberapa aplikasi, meskipun terkadang terdapat komponen yang sangat spesifik sehingga sulit untuk digunakan ulang pada aplikasi yang lain.

Komponen dapat diklasifikasikan dalam 2 kategori yaitu *simple* and *composite component*s. Jika kelakuan dari sebuah komponen dihasilkan dari sekumpulan *binary code*, maka komponen tersebut dikategorikan sebagai *simple component*.

Sedangkan, jika kelakuan dari komponen merupakan hasil penggabungan dari beberapa komponen, maka komponen tersebut dikategorikan sebagian *composite components*.

Composite components sering juga disebut dengan istilah *compound components*. Dalam makalah ini istilah komponen dikaitkan dengan *simple component*.

2.2 Komponen Terdistribusi

Komponen terdistribusi adalah komponen perangkat lunak yang disimpan di mesin yang terpisah dengan aplikasi klien dan difasilitasi oleh protokol komunikasi yang ditambahan kedalam komponen dasar. Dengan memisahkan komponen dari aplikasi klien diharapkan dapat memudahkan pengembang dalam melakukan perubahan fitur dan implementasi dari komponen di kemudian hari, tanpa harus mengkompilasi ulang aplikasi yang ada di klien.

Komponen-komponen terdistribusi biasanya dikelola oleh suatu kakas yang berfungsi untuk mengelola dan memonitor aktifitas dari komponen. Contoh dari kakas ini adalah Microsoft Transaction Server.

2.3 Aplikasi Terdistribusi

Arsitektur aplikasi adalah pandangan secara konseptual terhadap struktur dari sebuah aplikasi. Pada umumnya sebuah aplikasi terdiri dari kode program untuk pengolahan data*, business logic* dan *user interface*. Di dalam pengembangan aplikasi *enterprise* tradisional digunakan paradigma yang bersifat monolitik, artinya basis data*, business logic* dan *user interface* digabung kedalam satu aplikasi yang sama. Akibatnya ketika terjadi perubahan, seluruh sistem perlu dikompilasi ulang, sementara perubahan yang dilakukan tidak terlalu banyak. Kelemahan lainnya, *reusability* dari modul-modul sangat rendah. **[STA00]**

Jenis-jenis arsitektur aplikasi terdistribusi yang dikenal dalam lingkungan pengembangan perangkat lunak diantaranya adalah:

2-Tier Client-Server on Intranet
2-Tier Client-Server on Internet
3-Tier Client-Server on Intranet
3-Tier Client-Server on Internet

Saat ini aplikasi yang ada sebagian besar menggunakan arsitektur 2-tier atau lebih dikenal dengan aplikasi *client/server* yang dijalankan di intranet maupun internet. Pada arsitektur ini, klien menangani kode untuk pemrosesan data dan antarmuka dengan pemakai dari aplikasi. Sedangkan data disimpan dan dikelola oleh komputer *server*. Untuk mengakses data yang diperlukan maka komputer klien melakukan hubungan secara langsung ke komputer *server,* dan hubungan ini seringkali dilakukan selama aplikasi dijalankan.

Arsitektur 2-tier bekerja dengan baik pada lingkungan yang banyaknya pemakai terbatas, namun jika digunakan pada sistem dengan jumlah pemakai yang sangat banyak, kinerja dari sistem akan berkurang. Hal ini dikarenakan terbatasnya

ISBN: 979-756-061-6

hubungan yang tersedia antara Klien dan *Server*. Selain itu aplikasi yang ada pada klien relatif berukuran besar, karena setiap aplikasi di klien mengandung kode untuk pemrosesan data (sering disebut dengan istilah *fat client*). Repotnya lagi, jika kode untuk pemrosesan data memerlukan perubahan, maka aplikasi harus didistribusikan ulang ke seluruh komputer yang ada pada klien.

Kemudian timbul ide untuk melakukan sedikit perbaikan dengan cara memindahkan kode untuk pemrosesan data kepada komputer *server*. Misalnya dengan menggunakan fitur *Stored Procedures*, yang disediakan oleh beberapa DBMS Server. Arsitektur ini terkadang disebut dengan arsitektur "*two-and-a-half tier.*". Namun dari segi *scalability* dan *reuseability*, arsitektur ini masih memiliki keterbatasan. Masalah keterbatasan *scalability* dan *reuseability* ini akhirnya dapat diperbaiki secara siginifikan dengan diperkenalkannya arsitektur 3-tier. Pada model ini, kode untuk antarmuka dengan pemakai, pemrosesan data dan pengaksesan data dipisahkan satu dengan yang lainnya, namun perlu dicatat bahwa pemisahan yang dilakukan hanya pada sisi logik saja, bukan pemisahan secara fisik. Hebatnya lagi, kode-kode itu tidak perlu ditulis dalam bahasa yang sama, tidak perlu pula berada platform yang sama ataupun pada mesin yang sama. Arsitektur 3-tier dapat dilihat pada gambar berikut

Error! Not a valid link.
Arsitektur 3-Tier. [KIR98]

Dengan menggunakan model arsitektur 3-tier , maka sumber daya yang ada misalnya koneksi ke basis data dapat digunakan bersama oleh beberapa klien. Pada arsitektur ini klien tidak lagi berhubungan secara langsung dengan *data server,* karena komunikasi untuk melakukan permintaan data dilakukan melalui *business service*. Hal ini berbeda dari komunikasi klien dan *database* pada aplikasi 2-tier.

Pada aplikasi yang menggunakan arsitektur 3-tier, kode untuk antarmuka dengan pemakai, pemrosesan data dan pengaksesan data dipisahkan satu dengan yang lainnya. Karena adanya pemisahan ini, maka mulailah diperkenalkan istilah *Service Model*.

Service Model adalah cara yang digunakan untuk mengelompokkan komponen-komponen yang dibuat dalam sebuah aplikasi yang menggunakan arsitektur 3-tier. Ada tiga *service model* yang dikenal, yaitu: **[JER99]**

User Services / Presentation Layer
Business Services / Business Layer
Data Services / Data Access Layer

2.4 Keterkaitan Komponen dan Aplikasi Terdistribusi

Komponen tidak hanya dapat digunakan untuk mengembangkan aplikasi terdistribusi. Dalam pengembangan aplikasi tidak terdistribusi, pengembang perangkat lunak dapat menggunakan komponen sebagai bagian dari aplikasi yang dikembangkan. Biasanya komponen yang digunakan adalah komponen untuk *user interface* dan *data access.* Contohnya adalah pada aplikasi yang dikembangkan dengan menggunakan Delphi, dimana untuk keperluan masukan dan keluaran dari aplikasi, pengembang aplikasi yang menggunakan Delphi menggunakan komponen-komponen yang sudah disediakan oleh Borland dan pengembang lain yang mengembangkan komponen-komponen untuk dapat digunakan pada Delphi. Sedangkan pengembang yang menggunakan Visual Basic, menggunakan kontrol-kontrol untuk mengembangkan aplikasi terutama untuk *user interface* dan *data access.* Pada dasarnya kontrol adalah komponen yang dibangun dengan menggunakan teknologi **COM** yang disimpan dalam sebuah file yang memiliki ekstensi **OCX**.

Terdapat beberapa pengertian tentang aplikasi terdistribusi. Pengertian yang ***pertama*** adalah adanya pengembang yang memiliki pandangan bahwa jika basis data yang dikelola oleh aplikasi dipisahkan dari aplikasi dengan cara menyimpannya pada mesin yang lain (1-*tier application*) maka aplikasi tersebut dapat dikategorikan sebagai sebuah aplikasi terdistribusi. Pengertian yang ***kedua*** adalah adanya pengembang yang memiliki pandangan bahwa jika basis data dikelola oleh DBMS Server, sedangkan aplikasi hanya terdiri dari *User Interface* dan *Business Services* (*2-tier application*) maka aplikasi tersebut dapat dikategorikan sebagai sebuah aplikasi terdistribusi. Pengertian yang ***ketiga*** adalah adanya pengembang yang memiliki pandangan bahwa sebuah aplikasi dikategorikan sebagai aplikasi terdistribusi, jika kode untuk *user interfaces, business services,* dan *data access* dipisahkan satu dengan yang lainnya.

Dalam makalah ini, penulis sependapat dengan pengertian yang ***ketiga***. Sehingga hubungan antara komponen dan aplikasi terdistribusi ditekankan pada penggunaan komponen yang berada di lapisan *Business Services.* Dalam pengembangan aplikasi terdistribusi berbasis komponen, komponen untuk *business services* ini sering disebut dengan istilah *middleware*.

3. Studi Kasus

Untuk menerapkan teknologi pengembangan perangkat lunak berbasis komponen terdistribusi yang menjadi kajian dalam penulisan makalah ini, maka diperlukan suatu implementasi dalam bentuk studi kasus. Dalam memilih studi kasus yang akan

ISBN: 979-756-061-6

dibuat, penulis memutuskan untuk mengembangkan ulang aplikasi yang pernah dikembangkan oleh penulis. Dari sekian banyak aplikasi yang pernah dikembangkan oleh penulis, maka penulis memilih untuk mengembangkan ulang (*Reengineering*) aplikasi Sistem Informasi Akademik (**SIMAK**) di STMIK Mardira Indonesia yang dikembangkan dengan arsitektur 2-tier dan metodologi pengembangan konvensional, menjadi Distributed Sistem Informasi Akademik (**DISTSIMAK**) yang menggunakan arsitektur 3-tier. Alasan yang mendasari dilakukannya *reengineering* terhadap aplikasi **SIMAK**, adalah:

1. Dikaitkan dengan karakteristik teknologi pengembangan perangkat lunak berbasis komponen maka **SIMAK** cocok untuk dikembangkan ulang dengan menggunakan teknologi tersebut.
   a. Aplikasi-aplikasi yang ada pada sebuah *enterprise,* dapat menggunakan komponen yang sama tanpa harus mengkompilasi ulang aplikasi jika terjadi perubahan implementasi pada komopnen yang digunakan. Dengan mengembangkan ulang **SIMAK** melalui metodologi berbasis komponen maka akan komponen-komponen yang dibuat oleh aplikasi **SIMAK** akan dapat digunakan oleh aplikasi lain di lingkungan STMIK Mardira Indonesia.
   b. Karena komponen menjadi bagian utama dari perangkat lunak, maka jika dilakukan perubahan pada aturan bisnis, tidak perlu dilakukan perubahan pada aplikasi-aplikasi yang ada, melainkan cukup dilakukan pada komponen yang membentuk aplikasi tersebut.
   c. Dengan digunakannya komponen, maka aplikasi pada klien akan menjadi semakin tipis (*thin client*) yang mengakibatkan tidak diperlukannya sumber daya komputer yang terlalu tinggi.
2. Terdapat beberapa kelemahan dari arsitektur aplikasi **SIMAK,** diantaranya:
   a. *Business logic* pada **SIMAK** masih digabungkan kedalam *presentation logic.*
   b. Tidak adanya dokumentasi perangkat lunak **SIMAK**, mengakibatkan kesulitan saat dilakukan pemeliharaan terhadap aplikasi **SIMAK**.
   c. Perlunya intalasi ulang pada semua klien saat dilakukan perubahan.
   d. Tingginya sumber daya komputer yang dibutuhkan oleh setiap klien karena semua kode program dibebankan terhadap komputer klien.

Dengan demikian dalam pembuatan studi kasus pada makalah ini akan dilakukan pekerjaan-pekerjaan berikut:

- Melakukan tinjauan terhadap aplikasi **SIMAK.**
- Melakukan ***ReEngineering*** aplikasi **SIMAK** pada aplikasi **DISTSIMAK.**

Sebelum dilakukan pembahasan mengenai pekerjaan-pekerjaan tersebut, terlebih dahulu akan dilakukan perbandingan terhadap aplikasi **SIMAK** dan **DISTSIMAK** yang dilakukan pada beberapa sisi**.** Hal ini dilakukan untuk memperjelas perbedaaan maupun perubahan yang terjadi dari ketiga aplikasi tersebut. Perbandingan tersebut dapat dilihat pada berikut ini.

Perbandingan SIMAK dan DISTSIMAK

No	Aspek	SIMAK	DISTSIMAK
1	Arsitektur	2-tier	3-tier
2	Paradigma	*Service Model*	*Service Model*
3	Model Proses	Aliran Data	**CBD** (Component Based Development)
4	Metodologi	*Data Oriented*	*Object Oriented*
5	DBMS	Microsoft Access	Sybase
6	Platform	Windows Family	Windows Family
7	Tipe Aplikasi	Windows Based	Windows dan WEB Based
8	Aksi terhadap Komponen	*Use*	*Use*, dan *Create*
9	Model Komponen	COM	DCOM
10	Bahasa Pemrograman	VB dan Delphi	VB, ASP, dan Delphi
11	Kakas Pemodelan	-	Power Designer dan Rational Rose
12	Notasi Pemodelan	-	UML

3.1 Tinjauan Terhadap Aplikasi SIMAK

Tinjauan terhadap aplikasi **SIMAK** dilakukan dengan tujuan untuk mengetahui beberapa hal yang terkait dengan **SIMAK** ditinjau dari berbagai aspek**,** yaitu:

- **Spesifikasi**
  Aspek ini perlu ditinjau ulang dengan tujuan agar penulis dapat menentukan spesifikasi masukan,proses dan keluaran yang harus ada pada **DISTSIMAK.**
- **Arsitektur**
  Aspek ini perlu ditinjau ulang dengan tujuan agar dapat dilihat perbedaan arsistektur antara **SIMAK** dan **DISTSIMAK.**
- **Basis Data**
  Aspek ini perlu ditinjau ulang dengan tujuan untuk dapat dijadikan sebagain acuan untuk menentukan basis data yang harus ada pada DISTSIMAK.

ISBN: 979-756-061-6

3.1.1 Spesifikasi SIMAK

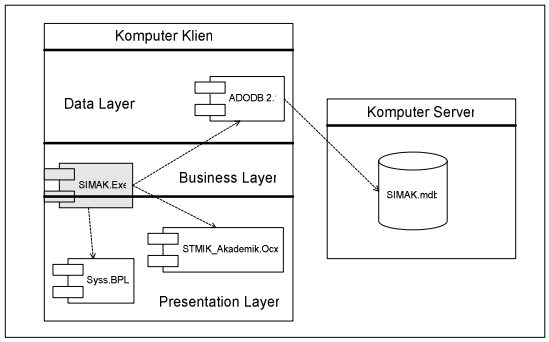
Masukan dari **SIMAK** adalah data akademik dan keuangan mahasiswa yang diperlukan dan dimasukkan oleh operator aplikasi **SIMAK**. Data yang dimasukkan terdiri dari data mahasiswa, dosen, kurikulum, registrasi, pembayaran uang kuliah , nilai dan lain-lain.

Proses yang dilakukan adalah melakukan proses penghitungan dan rekapitulasi data akademik yang telah dimasukkan. Proses yang terdapat dalam perangkat lunak ini diantaranya penghitungan IPK, pembuatan transkrip, merekap jumlah mahasiswa yang melakukan registrasi dan lain-lain.

Keluaran perangkat lunak SIMAK adalah berupa informasi data akademik, misalnya Kartu Hasil Studi, Kartu Studi Mahasiswa dan lain-lain.

3.1.2 Arsitektur SIMAK

SIMAK diimplementasikan dengan menggunakan arsitektur 2-tier dengan mode *multi-user*. Aplikasi SIMAK terdiri atas beberapa file eksekusi yang berhubungan langsung dengan pengguna dan berisi modul-modul seperti diuraikan pada 4.1.1. File eksekusi ini menggunakan komponen dan paket (lihat bagian 4.1.4) yang melakukan perubahan terhadap data yang disimpan pada file basis data yang disimpan pada komputer server. Basis data *SIMAK* dikelola oleh DBMS Microsoft Access. Arsitektur dari SIMAK dapat dilihat pada gambar berikut.



Arsitektur Aplikasi SIMAK

SIMAK menggabungkan seluruh layanan aplikasi yang terbagi atas beberapa modul dalam satu wadah, sehingga aplikasi SIMAK bersifat *fat client*. Untuk mengimplementasikan seluruh layanan tersebut telah digunakan beberapa komponen dan kontrol yang disediakan oleh bahasa pemrograman Delphi dan VB, maupun komponen dan kontrol yang dibuat sendiri oleh penulis.

3.1.3 Basis Data SIMAK

Dalam melakukan tinjauan terhadap basis data SIMAK *yang* menggunakan DBMS Microsoft Access, dilakukan proses *reverse engineering* pada basis data SIMAK dengan menggunakan kakas Power Designer. Hasilnya diketahui bahwa basis data SIMAK terdiri dari beberapa tabel, yaitu:

Absensi_Dosen	Agama
Bagian	Bayar_Per_Semester
Bayar_Per_Tahun	Bayar_Sekali
Biaya	Biodata_Dosen
Biodata_Mahasiswa	Biodata_Staff
Datum	Datum1
Dosen	Extra_Jadwal
Golongan_Darah	Hak_Akses
Hari	IPK
Item_Kwitansi	Jadwal
Jadwal_UAS	Jadwal_UTS
Jam	Jenis_Kelamin
Jenis_Kwitansi	Jenis_Mata_Kuliah
Jurusan	Kelas
Keterangan_Mata_Kuliah	Kewarganegaraan
KHS	KHS_Pindahan
Kota	Kwitansi
Lembaga	Log_Book
Mahasiswa	Mata_Kuliah
Nilai	Orangtua
Pekerjaan	Pemakai
Pendidikan	Rekap_Nilai
Ruangan	Semester
Staff	Status_Dosen
Tahun_Akademik	Transfer
Transkip	Uraian_Kwitansi

3.2 Reengineering SIMAK menjadi DISTSIMAK

Tahapan *Reengineering* SIMAK menjadi **DISTSIMAK** dibagi kedalam 2 bagian, yaitu:

- Analisis dan Perancangan **DISTSIMAK**
- Analisis komponen **DISTSIMAKServer** dan perancangan *interface* beserta kelas-kelas yang mengimplementasikan kode dari beberapa *interface* yang ada pada **DISTSIMAKServer**, dimana komponen **DISTSIMAKServer** merupakan bagian penting dari prototipe aplikasi **DISTSIMAK**.

3.2.1 Analisis dan Perancangan DISTSIMAK

Pada makalah ini, untuk tahapan analisis, diagram UML yang digunakan adalah Use Case Diagram. Use Case Diagram digunakan dengan tujuan untuk memodelkan kelakuan sistem, dan juga untuk menggambarkan *requirement* perangkat lunak. Di dalam Use Case Diagram digambarkan aktor, *use case*, dan relasi keduanya. *Use case* adalah deskripsi aksi yang dilakukan sistem. Aktor menggambarkan peran yang dimainkan pengguna *use case* pada saat melakukan interaksi dengan *use case*. Use Case Diagram dan penjelasan selengkapnya dapat dilihat pada **SPL_DISTSIMAK**.

Diagram lain yang digunakan adalah **Object Diagram**, **Class Diagram**, **Component Diagram** dan **Sequence Diagram** yang menggambarkan

ISBN: 979-756-061-6

interaksi antar obyek dan relasinya. Diagram ini merupakan aspek dinamis dari sistem. **Sequence Diagram** ditunjukkan pada **SPL-DISTSIMAK**.

Semua diagram yang digunakan dalam makalah ini menggunakan tool Rational Rose, yang sekaligus dapat digunakan untuk menguji diagram yang dibuat.

Secara umum, fungsi dari perangkat lunak **DISTSIMAK** adalah untuk menerapkan teknologi pengembangan perangkat lunak berbasis komponen. Selain itu perangkat lunak ini dapat digunakan untuk membantu pengelolaan administrasi akademik di STMIK Mardira Indonesia, meskipun untuk saat ini fitur yang disediakan masih dibatasi.

Fungsi-fungsi yang dimiliki perangkat lunak **DISTSIMAK**, secara umum adalah:

- Pengelolaan dan pemrosesan data yang menjadi entitas utama dari sistem akademik.
- Pengelolaan dan pemrosesan data kurikulum.
- Pengelolaan dan pemrosesan data registrasi.
- Pengelolaan dan pemrosesan data nilai.

Dalam pengembangan studi kasus **DISTSIMAK** akan digunakan teknologi berbasis komponen. Model komponen yang akan digunakan adalah DCOM, arsitektur yang digunakan *3-tier client/server intranet*, untuk memodelkan sistem akan digunakan bahasa UML, sedangkan dalam implementasi perangkat lunak akan digunakan bahasa pemrograman Delphi dan Visual Basic.

Dipilihnya kedua bahasa ini didasari kepada pertimbangan bahwa kedua bahasa tersebut memberikan dukungan penuh terhadap seluruh aspek yang digunakan dalam mengembangkan studi kasus **DISTSIMAK**. Dukungan tersebut berupa:

- Dukungan terhadap pembuatan program dengan arsitektur 3-tier.
- Dukungan terhadap konsep *software reuse.*
- Visual Basic didukung oleh kakas *Rational Rose,* sehingga dapat dilakukan integrasi antara model yang dibuat dengan bahasa UML dengan program yang dibuat dengan Visual Basic.
- Dukungan penuh terhadap teknologi DCOM.

3.2.2 Hasil yang Diharapkan

Dengan adanya perubahan arsitektur dari 2-tier ke 3-tier, diharapkan perangkat lunak sebagai pengganti dari **SIMAK** dapat memenuhi hal-hal berikut:

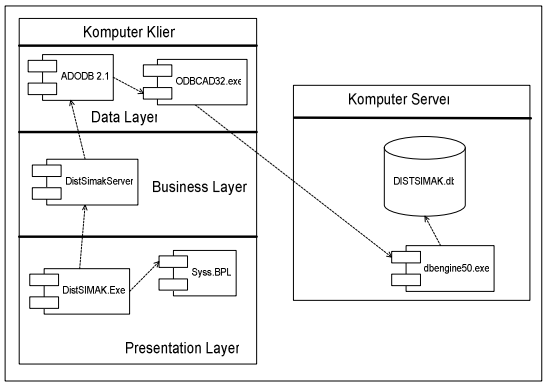
- Dengan digunakannya komponen **DISTSIMAKServer** maka fungsi bisnis dari **DISTSIMAK** dapat diimplementasikan dengan mudah.
- Kemudahan dalam melakukan evolusi sistem melalui penggunaan komponen untuk *server-side* yang membungkus aplikasi dan data.
- Kemudahan dalam modularitas dan interoperabilitas dengan aplikasi lain melalui *componentization* seluruh aplikasi.

3.2.3 Arsitektur DISTSIMAK

DISTSIMAK diimplementasikan dengan menggunakan arsitektur 3-tier pada jaringan Intranet. Alasan pemilihan arsitektur 3-tier karena arsitektur ini mendukung konsep reuse, hal ini dirasakan perlu karena:

- Aplikasi yang baru, dikembangkan pada level tertinggi dibandingkan dengan data sederhana dan kelas-kelas untuk string.
- Penggunaan ulang dari *Business Components* mempengaruhi investasi pada sisi server dalam aspek arsitektur dan pemodelan.

Aplikasi **DISTSIMAK** terdiri file eksekusi yang berhubungan langsung dengan operator **DISTSIMAK** dan berisi modul-modul. File eksekusi ini menggunakan komponen dan paket yang melakukan perubahan terhadap data yang disimpan pada file basis data yang disimpan pada komputer server. Basis data **DISTSIMAK** dikelola oleh DBMS Sybase SQL Server.



Arsitektur Distributed SIMAK

3.2.4 Perancangan DISTSIMAK

Untuk melakukan perancangan perangkat lunak **DISTSIMAK** yang akan dibuat, digunakan diagram **UML** yaitu **Class Diagram**, **Component Diagram**, dan **Sequence Diagram**. Penggunaan diagram dan penjelasannya secara terperinci dapat dilihat pada **SPL-DISTSIMAK**. Bagian ini hanya mengemukakan perancangan perangkat lunak secara global.

Tahapan yang diambil dalam perancangan perangkat lunak **DISTSIMAK** adalah:

1. Perancangan masukan dan keluaran
2. Perancangan aspek statis perangkat lunak, yang meliputi penentuan struktur data internal, penentuan kelas dan hubungan antar kelas yang terjadi. Diagram yang digunakan adalah **class diagram**.
3. Perancangan aspek dinamis, yang meliputi penghidupan obyek, pemanggilan layanan yang disediakan obyek, dan pemusnahan obyek. Diagram yang digunakan adalah **collaboration diagram** dan **sequence diagram**.

ISBN: 979-756-061-6

a. Perancangan Masukan dan Keluaran

Perancangan masukan dan keluaran meliputi masukan yang diberikan pengguna, sedangkan keluarannya adalah keluaran secara visual dan keluaran yang berbentuk *report* yang dapat ditampilkan pada layar monitor, maupun dicetak ke kertas melalui printer.

b. Perancangan Aspek Statis

Pada perancangan aspek statis, yang dikemukakan adalah struktur data internal yang digunakan untuk menyimpan informasi, serta kelas yang diperlukan oleh **DISTSIMAK**.

Untuk menyimpan semua infomasi yang ada pada perangkat lunak **DISTSIMAK** digunakan objek-objek yang merupakan *instance* dari kelas-kelas yang ada pada komponen yang digunakan oleh perangkat lunak **DISTSIMAK**. Gambar berikut menunjukkan **Object Diagram** pada perangkat lunak **DISTSIMAK.**

c. Perancangan Aspek Dinamis

Pada perancangan aspek dinamis, yang diketengahkan adalah pengorganisasian obyek dan interaksi yang terjadi antar objek.

3.2.5 Pembuatan Komponen DISTSIMAKSERVER

Pada makalah ini akan dibuat komponen **DISTSIMAKServer** yang dapat digunakan dalam lingkungan komputasi terdistribusi khususnya untuk aplikasi **DISTSIMAK** yang menggunakan arsitektur 3-tier.

Model komponen yang digunakan oleh penulis adalah DCOM. Alasan-alasan yang mendasari digunakannya model komponen ini adalah:

a. Bahasa pemrograman yang digunakan saat implementasi adalah Delphi dan VB. Kedua bahasa ini memberikan dukungan penuh terhadap teknologi DCOM (lebih jelasnya lihat bagian lampiran). Apalagi DCOM dan VB dikembangkan oleh *vendor* yang sama yaitu Microsoft, sehingga interaksi diantara keduanya sangatlah erat.
b. Platform yang digunakan oleh aplikasi yang dibuat adalah keluarga dari Windows, dengan demikian penggunaan DCOM untuk digunakan pada platform menjadi mudah.
c. Aspek usabilitas, *robustness, simple instalation,* dan *integration* yang menjadi fitur utama yang harus dimiliki perangkat lunak saat ini dapat terpenuhi dipenuhi oleh point a,b.
d. Properti yang harus dimiliki oleh sebuah komponen terdistribusi seperti diuraikan pada bab II, semuanya dimiliki oleh DCOM meskipun untuk beberapa properti seperti dukungan terhadap *cross-platform, Common Services, Transparently* tidak terlalu baik, namun properti-properti ini tidak menjadi kebutuhan utama dari aplikasi yang dikembangkan sebagai studi kasus.
e. Dalam pembuatan komponen-komponen yang akan digunakan dalam studi kasus, maka model komponen yang akan digunakan harus dapat dapat memberikan solusi terhadap permasalahan berikut:
   - ***Basic Interoperability***
     Bagaimana suatu komponen yang dikembangkan oleh suatu pengembang dapat diyakinkan untuk dapat *interoprate* dengan komponen lain yang dikembangkan oleh pengembang yang berbeda.
   - ***Versioning***
     Bagaimana mengupgrade suatu komponen tanpa memerlukan upgrade pada komponen lain pada suatu sistem?.
   - ***Language independence***
     Bagaimana komponen-komponen yang ditulis dalam bahasa yang berbeda dapat saling berkomunikasi?.
   - ***Transparent cross-process interoperability***
     Bagaimana memberikan keleluasaan kepada pengembang untuk membuat suatu komponen untuk dapat dijalankan dalam mode *in-process* atau *cross-process*, dengan menggunakan model pemrograman yang sederhana.

Dari hasil eksplorasi yang dilakukan penulis terhadap model komponen DCOM, penulis menemukan fakta bahwa fitur-fitur yang disediakan DCOM sangat mendukung untuk memecahkan masalah tersebut. Adapun fitur-fitur yang dimiliki DCOM adalah :

- Komunikasi antar komponen, baik antar proses maupun dalam batasan
- Manajemen memori secara bersama antar komponen
- Pelaporan status dan kesalahan
- Pemuatan komponen secara dinamis.

4. Kesimpulan

Dari uraian diatas dapat disimpulkan bahwa untuk membangun perangkat lunak berbasis komponen, diperlukan syarat-syarat sebagai berikut :

a. Pemahaman terhadap teknologi pengembangan perangkat lunak berbasis komponen .
b. Pengetahuan tentang kemampuan dari beberapa *tools* yang mendukung teknologi yang dikaji.
c. Pembuatan komponen-komponen terdistribusi , yang dapat digunakan oleh pemrogram yang mengembangkan perangkat lunak berbasis komponen terdistribusi.

Daftar Pustaka

[BAS98] Bass L., Clements P., Kazman R.Britton, *Software Architectures in Practice.* Addison-Wesley, 1998

ISBN: 979-756-061-6

[BRI01] Britton, Chris. *IT Architectures and Middleware.* Addison-Wesley, 2001

[BRO90] Brock, Rebecca., Wiener, Lauren., Wilkerson, Brian., *Designing Object Oriented Software*. Prentice-Hall. 1990.

[BRO00] Brown A., *Large-scale Component-based Development.* Prentice Hall, 2000

[BOS00] Bosch J., *Design and Use of Software Architectures.* Addison-Wesley, 2000

[CAN99] Cantu, Marco. *Mastering Delphi 5.*Sybex, 1999

[COA91] Coad, Peter., Yourdon, Edward., *Object Oriented Design*. Prentice-Hall. 1991.

[COA93] Coad, Peter., Nicola,Jill., *Object Oriented Programming*. Prentice-Hall. 1993.

[CRN00a] Crnkovic Ivica.,Larsson Magnus , *New Paradigm of Software Development.* Addison-Wesley, 2000

[CRN00b] Crnkovic I., Larsson M., Küster Filipe J. K., Lau K., *Databases and Information Systems, Fourth International Baltic Workshop, Baltic DB&IS*, Selected papers, Kluwer Academic Publishers 2001, pp.237-252

[CRN97] Crnkovic, Ivica., Larsson, Magnus., *Component-Base Software Engineering.* Microsoft Press, 1997

[HEI01] Heineman G. and Councill *W. Component-based Software Engineering, Putting the Pieces Together*. Addison Wesley, 2001

[HEN97] Henderson, Ken. *Client/Server Developer's Guide With Delphi 3.* Sams Publishing, 1997

[JER99] Jerke, Noel, et. al. *Visual Basic 6 Client/Server How-To*. Sams Publishing, 1999

[KIR98] Kirtland, Mary. *Designing Component-Based Applications.* Microsoft Press, 1998

[KOT01] Kotonya G. and Rashid A., A strategy for Managing Risks in Component-based Software Development,*27 th Euromicro Conference 2001 Proceedings*, IEEE Computer society, 2001, pp. 12-21

[LIB99] Liberty, Jesse, et. al. *Object Oriented Analysis and Design.* Sams Publishing, 1999

[MOR01] Morris J., Lee G., Parker K., Bundell G., Peng Lam C., "Software Component "Certification", *IEEE Computer, 200*1, September

[MRV97] Mrva M., *Reuse Factors in Embedded System Design*, High-Level Design Techniques Dept. At Siemens AG, 1997

[OPC03] OPC Foundation, http://www.opcfoundation.org/, Access Date 2003-04-17

[PRE97] Pressman, Roger. *Software Engineering.* McGraw-Hill, 1997

[SCH99] Schmuler, Joseph. *Teach Yourself UML in 24 Hours.* Sams Publishing, 1999

[STA00] Stamatakis, William. *Microsoft Visual Basic Design Patterns.* Microsoft Press, 2000

[WAL01] Wallnau K. and Stafford J., Ensembles: Abstractions for a New Class of Design Problem, *27 th Euromicro Conference 2001 Proceedings*, IEEE Computer society, 2001, pp. 48-55

[XML03] Extensible Markup language(XML), http://www.w3.org/XML , Access Date 2003-05-27

[VOA00] Voas J. and Payne J., Dependability Certification of Software Components, *Journal of Systems and Softwa*re, No. 52, 2000, pp. 165-172.