Implementasi Multithreading Java pada Sistem Kolaborasi (Studi kasus Ruang Diskusi Maya)

Jasman Pardede, M.T [1)]

[1)] Teknik Informatika Fakultas Teknologi Industri, Institut Teknologi Nasional
Jl. PKH. Hasan Mustapa No.23, Bandung 40124 Indonesia
email : pardede_js@yahoo.com

ABSTRACT

Collaborating in completing task in a group is a important part in business process, where everyone needs to discuss his ideas, to share his opinion, to coordinate his planning, to make next worked planning and makes a concluding together. The increasing mobility of humans made a member of group can't ever the same place and time so that very hard to collaborate. Therefore needs tools to facility collaborate without time and place considered. Software application as tools collaborate is called groupware.

Groupware can be good applied needs a server application. The server application is responsible to manage and facility each client to collaborate with controlling activity each user, managing session and workspace by preparing functionality of text and image sending in the group. The responsibility of server application can run by well, it must capable run more than one calling the same method i.e. processing server must make "cloning itself". The each of the result processing server will interact to client. To implementation of this idea, researcher will implement multithreading java by case study room meeting virtual class.

Key words

collaborating, server, client, multithreading, java

1. Pendahuluan

Kolaborasi merupakan suatu kegiatan yang memungkinkan sekelompok orang bekerja dan berkomunikasi secara bersama-sama dan mempunyai sekumpulan aturan atau kebijakan tertentu yang berlaku pada setiap orang dalam kelompok tersebut [1].

Tingkat mobilitas manusia yang semakin tinggi, sangat membutuhkan adanya suatu wadah yang memfasilitasi kolaborasi pada waktu dan lokasi yang berbeda-beda[3]. Perangkat lunak yang mewadahi sistem kolaborasi disebut *groupware* [5]. Agar *groupware* berjalan dengan baik perlu adanya suatu aplikasi server yang bertanggung jawab mengatur dan memfasilitasi setiap client yang berkolaborasi, dengan cara "menggandakan dirinya" pada setiap akan berinteraksi dengan client.

Pengembangan aplikasi sistem kolaborasi tidak terlepas dari dukungan teknologi pemrograman jaringan, salah satunya adalah dengan pemrograman java *socket***.** Bahasa pemrograman java sejak pertama kali diluncurkan telah menyatakan diri sebagai bahasa yang *general purpose*, berorientasi objek, dan konkuren [4].

Konkurensi pada java didasarkan atas mesin *virtual* java yang mendukung banyak *thread* pada suatu saat. Setiap *thread* secara otonom mengeksekusi kode-kode java baik di dalam mesin pemroses tunggal maupun banyak mesin pemroses[6]. Setiap *thread* yang diciptakan pada java memiliki memori kerja (*working memory*) yang digunakan untuk menyimpan *copy* dari variabel-variabel yang digunakan. Ketika suatu *thread* mengeksekusi program java, *thread* melakukan operasi pada memori kerja, sementara memori utama digunakan sebagai *master copy*. Dukungan pengeksekusian banyak *thread* pada aplikasi yang sama pada waktu yang sama yang disebut multithreading akan menfasilitasi setiap pengguna dalam berkolaborasi.

Penelitian ini bertujuan untuk melakukan analisis, perancangan, dan pengimplementasian multithreading java pada sistem kolaborasi berbasis *web* yang memenuhi kebutuhan dalam mendukung sistem kolaborasi pada suatu *website*.

2. Studi Konsep Sistem Kolaborasi

Pengembangan aplikasi perangkat lunak yang dapat memfasilitasi kolaborasi perlu memperhatikan beberapa hal penting yaitu kebijakan dalam sistem kolaborasi,

komunikasi kelompok, dukungan teknologi pemrograman jaringan dalam mendukung pengembangan aplikasi perangkat lunak sistem kolaborasi, serta bagaimana aplikasi server dapat mengatur sistem dalam berkolaborasi dengan menerapkan konsep multithreading.

2.1 Kebijakan dalam Sistem Kolaborasi

Secara umum terdapat empat mode pengkoordinasian aktivitas-aktivitas yaitu; *parallel*, *pooled*, *sequential*, dan *reciprocal*. Sitiap aplikasi sistem kolaborasi harus mendukung satu atau lebih mode pengkoordinasian aktivitas [8].

2.2 Komunikasi Kelompok

Komunikasi kelompok adalah komunikasi dengan pertukaran pesan di mana pesan dikirimkan oleh seorang anggota kelompok dan diterima oleh seluruh anggota kelompok. Terdapat beberapa cara/teknik untuk mengimplementasikan komunikasi kelompok dalam mendukung sistem kolaborasi, yaitu Pemrograman Socket dengan TCP/IP atau UDP/IP, *Remote Procedure Call* (*RPC*) dan Java RMI [2].

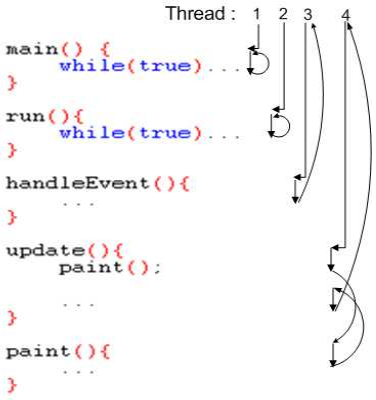
2.3 Multithreading

Thread merupakan suatu aspek penting bagi pemrograman java. Thread memberikan penulisan yang efisien dan server yang *robust* dalam memproses setiap client yang terhubung dalam thread yang terpisah.

Konsep multithreading adalah memberikan dan memperbolehkan banyak thread dieksekusi pada program yang sama pada waktu yang bersamaan, seperti yang ditunjukkan pada Gambar 1. Ini tidak sama dengan multitasking, karena thread-thread mengeksekusi pada data space yang sama. Jika suatu variabel global diubah pada suatu thread maka semua thread akan memperhatikan perubahan tersebut[7].

Thread pada java diciptakan dan dikelola melalui kelas Thread yang terdapat di package standar **java.lang**. Pada java, menciptakan suatu objek kelas Thread akan menciptakan suatu thread [6].

Kelas Thread di java menyediakan sejumlah metode untuk mengendalikan thread misalnya untuk memasuki antrian (start), istirahat (sleep), dan lain-lain. Metode dari kelas Thread yang harus didefinisikan kembali oleh sub kelas adalah metode run. Metode run akan dipanggil secara otomatis saat suatu thread memegang kendali pemroses. Selain kelas Thread, java juga menyediakan interface Runnable yang dapat diimplementasikan oleh kelas yang mengimplentasikan metode run[7].



Gambar 1 Aplikasi multithreading

Setiap thread memiliki prioritas yang menentukan penjadwalan thread. Bila thread diciptakan tanpa mendefinisikan prioritas secara eksplisit maka akan digunakan prioritas default, yaitu mempunyai nilai prioritas lima. Bila terdapat suatu thread yang memiliki prioritas yang lebih tinggi dari thread yang lain maka thread tersebut akan segera dijalankan sedangkan thread yang sebelumnya sedang berjalam akan segera menuju antrian (preemptive). Dengan mendefinisikan atribut *modifier* sebagai *volatile* akan dijamin bahwa setiap akses ke atribut tersebut yang akan dilakukan oleh suatu *thread* dijamin akan mengakses nilai *current* pada memori utama.

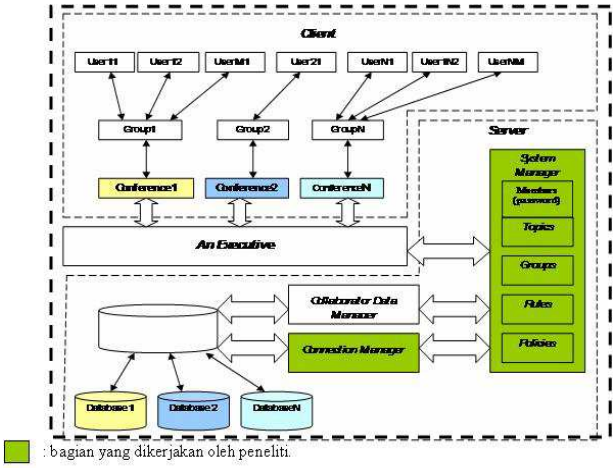
3. Hasil Percobaan

Studi kasus penelitian ini adalah sebuah ruang kelas pertemuan maya. Pada sebuah ruang kelas terdapat seorang pengajar dan beberapa siswa. Pengajar yang mengontrol satu ruangan disebut *initiator*. Siswa yang berperan aktif dan dikontrol oleh *initiator* disebut *participant*. Kelangsungan proses kolaborasi dikontrol oleh *moderator*. *Initiator* atau *participant* yang berada pada suatu ruang kelas disebut *collaborator*. Media kolaborasi setiap ruangan menggunakan sebuah *whiteboard* sebagai *drawing tools* dan *chat system* sebagai *presentation tools.* Gambaran umum dari perangkat lunak yang akan dibangun ditunjukkan pada Gambar 2.

Setiap *collaborator* yang mengikuti pembahasan topik yang sama akan dimasukkan dalam satu kelompok. Setiap *collaborator* akan memperoleh informasi yang sama dengan setiap pengguna dengan cara pengelompokkan *thread* pengguna menggunakan *ThreadGroup* sehingga *thread* lebih mudah dikontrol. Untuk mendukung suatu kelompok berkolaborasi lebih efektif digunakan bantuan sistem jaringan komputer (*networked computer system*) dan menerapkan kebijakan-kebijakan yang ditranslasikan kepada aturan-aturan (*rules*), yang disebut *coordination*

policies. Dari gambaran ini dapat dibuat arsitektur perangkat lunak sistem kolaborasi, seperti pada Gambar 3.

Gambar 2 Gambaran umum aplikasi

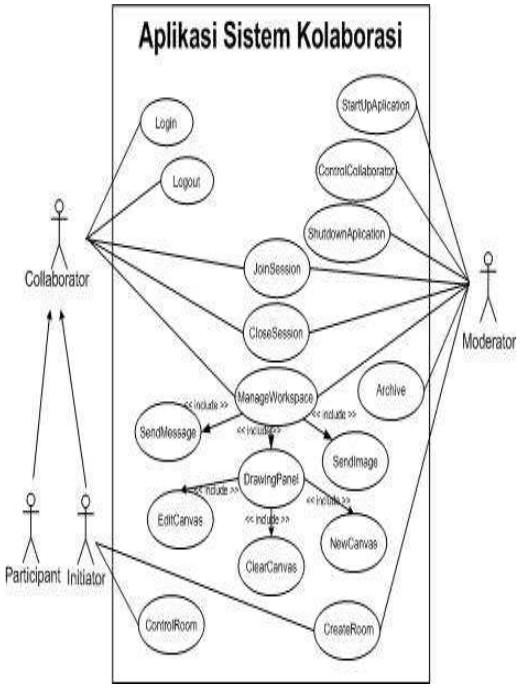


Gambar 3 Arsitektur perangkat lunak

Dari fungsional perangkat lunak yang hendak dibangun, diperoleh use case diagram seperti yang ditunjukkan pada Gambar 4.

Aplikasi yang dikembangkan pada penelitian ini menggunakan mode pengkoordinasian aktivitas secara *parallel mode*, di mana setiap pengguna dapat menggunakan *workspace*-nya tanpa adanya interupsi dari pengguna lain.

Dalam menerapkan *concurrency control* menggunakan konkurensi yang dimiliki java atas mesin *virtual* java yang mendukung banyak *thread* pada suatu saat, yang sering disebut dengan istilah *multithreading*. Untuk menerapkan *access control*, sistem akan mengidentifikasi setiap pengguna berdasarkan *users id* yang dimiliki oleh setiap pengguna. Pengguna yang berperan sebagai *initiator* akan mengatur hak akses setiap pengguna lainnya, apakah pengguna dapat menggambar pada *whiteboard* atau hanya dapat menulis pada *chat system*, hanya memperhatikan pengguna lain atau dapat menulis pada *chat system* dan menggambar pada *whiteboard*. Pengaturan hak akses ini dilakukan dengan pengiriman *token* ke pengguna oleh *initiator*.



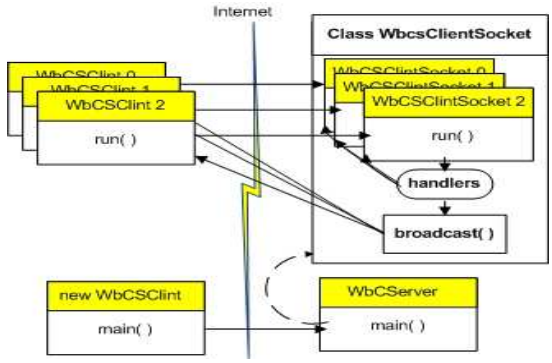
Gambar 4 Use case diagram

3.1 Perancanngan Sistem

Untuk menentukan letak penampilan pesan yang dikirim setiap pengguna dilakukan dengan cara menambahkan suatu label pada *header* setiap pesan, sehingga *whiteboard* dan *chat system* dapat beroperasi pada satu saluran komunikasi, seperti ditunjukkan pada Gambar 5.

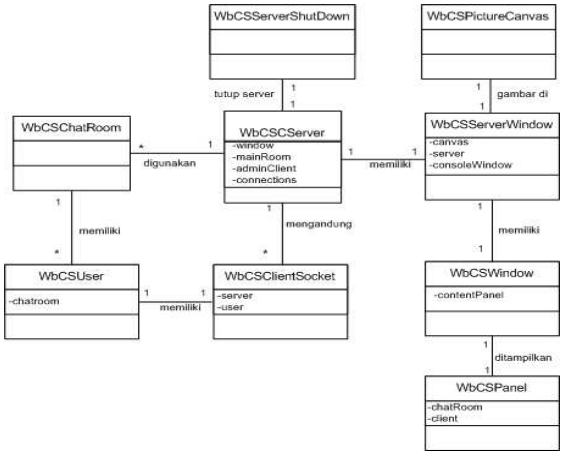
Gambar 5 Saluran komunikasi

Setiap client yang terhubung ke server akan diterima (*accept*) oleh kelas WbCSServer dan diberikan kepadanya suatu thread untuk mengontrol komunikasi dengan server. Kelas WbCSClientSocket secara aktual bekerja untuk mendengarkan pesan dan mengirimkan pesan ke setiap client dalam satu kelompok, seperti yang ditunjukkan pada Gambar 6.

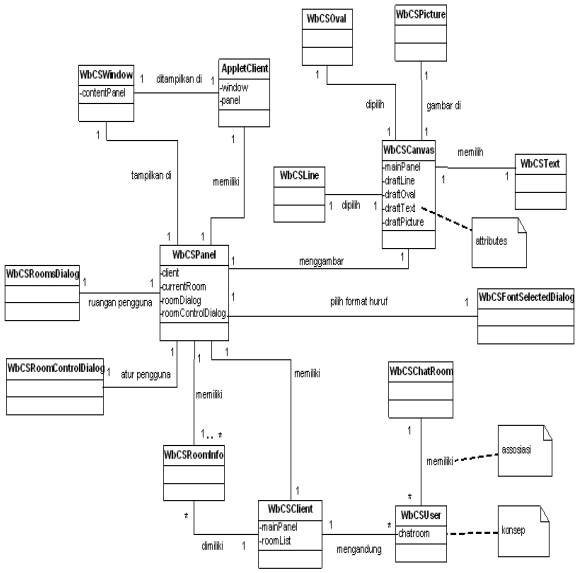


Gambar 6 Framework sistem kolaborasi

Model konseptual dari aplikasi dari sisi server digambarkan pada Gambar 7. Sedangkan model konseptual aplikasi dari sisi client digambarkan pada Gambar 8.



Gambar 7 Model konseptual pada sisi server



Gambar 8 Model konseptual pada sisi client

3.2 Implementasi Sistem

Perangkat lunak aplikasi, memerlukan dukungan perangkat lunak lain dalam implementasinya seperti, sistem operasi Windows atau Linux, bahasa pemrograman java serta *servlet engine* Apache Tomcat 5.5 sebagai *server side web application* berbasis java (JSP/servlet) dan MySQL Server 5.0 pada suatu server**.**

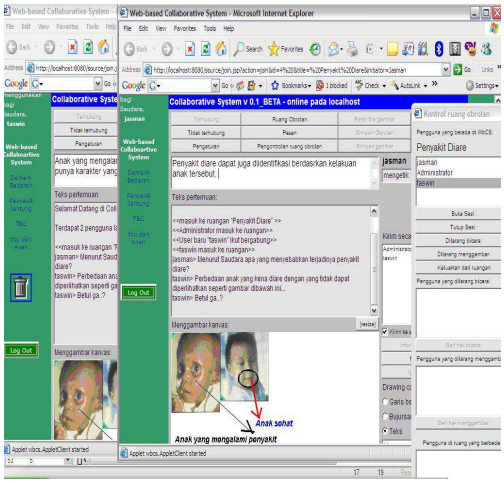
3.3 Teknik Pengujian

Teknik pengujian yang digunakan adalah teknik pengujian *black box testing*. Pengujian ini memungkinkan pemrogram untuk memperoleh sekumpulan kondisi masukan (*input*) yang akan secara penuh menjalankan semua kebutuhan fungsional untuk sebuah program. Dalam hal ini diambil salah satu butir uji, yaitu pengiriman pesan teks seperti yang ditunjukkan pada Tabel 1.

Tabel 1 Pengujian pengiriman pesan teks

Identifikasi	**TWbCS-01**		
Nama Butir Uji	Pengiriman Pesan Teks		
Tujuan	Media dalam mengirimkan pesan teks ke *workspace* pengguna dan menyimpan teks ke tabel		
Deskripsi	Pengguna menulis **string teks** ke teks field aplikasi kemudian tekan “ENTER”		
Kondisi Awal	Menu aplikasi applet menampilkan antar muka menu ruangan yang sedang dimasuki oleh pengguna		
Pengujian			
Skenario Uji			
1. Masukkan **string teks** yang akan dikirim 2. Pilih pengguna yang akan menerima pesan 3. Tekan “ENTER”			
Kriteria Evaluasi Hasil			
1. Pengguna mempunyai hak untuk mengirimkan teks 2. Pengguna yang dalam satu sesi tidak mengaktifkan tombol “Ignore user(s)”			
Kasus dan Hasil Uji (Data Normal)			
Masukan	Harapan	Pengamatan	Kesimplan
String teks	Tampilkan **string teks** ke *workspace*. Simpan **string teks** ke tabel.	**String teks** yang dikirim ditampilkan dan data disimpan di tabel	**[X]** Terima **[ ]** Tolak

Hasil pengujian pengiriman pesan teks dan gambar oleh pengguna direpresentasikan seperti pada Gambar 9.



Gambar 9 Menu teks area penulisan komentar dan pengiriman gambar

4. Kesimpulan

Berdasarkan penelitian yang dilakukan, beberapa kesimpulan yang dapat diambil sebagai berikut :

1. Pada penelitian ini telah mengimplementasikan multithreading java pada sebuah aplikasi server sebagai wadah sistem kolaborasi dalam mendukung pekerjaan yang membutuhkan kerja sama kelompok dengan memperhatikan aspek-aspek kelompok *awareness*, pengaturan komputer-komputer *client* yang terhubung ke server, pengaturan hak akses dan pemantauan aktivitas.
2. Konkurensi pada java didasarkan atas mesin *virtual* java yang mendukung banyak *thread* pada suatu saat, mampu mendukung komunikasi kelompok di dalam pengembangan aplikasi sistem kolaborasi dan dalam pengelompokkan *thread* dilakukan dengan menggunakan *ThreadGroup* sehingga *thread* dapat lebih mudah dikontrol.

REFERENSI

[1] Bocig, P., Chaffey, D., et al. (1999). *Business Information Systems Technology, development and Management*, Prentice-Hall.

[2] Coulouris, G., Dollimore, J., Kindberg, T. (2001). *Distributed Systems Concepts and Design*, 3th edition, Pearson Education.

[3] Geon-Tae Ahn, Jin-Hong Kim, Myung-Joon Lee. (2003). *A Web-based Collaborative System and Its Application.* School of Computer Engineering and Information Technology.

[4] Gosling, James et al. (1996). *The Java Language Specification***,** Sun Microsystems.

[5] Lynch, K.J., Snyder, J.M., Vogel, D.R. and McHenry, W.K., *The Arizona Analyst Information System : Supporting Collaborative Research on International Technological Trends***,** in Gibbs, S. And Verrijn-Stuart, A.A(eds): Multi-User Interfaces and Applications, North-Holland, Amsterdam, pp :159-174

[6] Mahmoud, Q.H.. (2000). *Distributed Programming with Java***,** Manning Publication Co.

[7] Merlin, Hughes, C., et al (1997). *Java Network Programming*, Manning Publications Co.

[8] Turoff, M. (1999). *An End to Student Segregation*: *No More Seperation Between Distance Learning and Regular Courses***:** A summary of the invited plenary for the Telelearning 99 meeting in Montreal, Canada.

Jasman Pardede, memperoleh gelar S.Si dari Universitas Andalas tahun 2001. Kemudian tahun 2005 memperoleh M.T bidang Rekayasa Perangkat Lunak dari Institut Teknologi Bandung, Indonesia. Saat ini sebagai dosen tetap program studi Teknik Informatika Institut Teknologi Nasional Bandung.