p-ISSN : 2460-7363
e-ISSN : 2614-6606

SIMULASI PEMBELAJARAN *CONCURRENCY* PADA MATA KULIAH SISTEM OPERASI BERBASIS MULTIMEDIA

[1]Erwin Yulianto [2]Galih Abdul Fatah Maulani
[1]Universitas Langlangbuana, Indonesia
[2]Universitas Garut, Indonesia
[1]rwinyulianto@yahoo.com
[2]galihafm@uniga.ac.id

Abstract : *The course of the operating system is one of the compulsory subjects that must be taught to students who take the majors / study program of informatics engineering / information systems and information technology. This course introduces students to the concept of the operating system, the types of operating systems that used in everyday life, the general services owned by the operating system, and to learn the techniques and algorithms used in the services management implementation of in the operating system. The operating system is a program that acts as an intermediary between the user, the application program and the hardware. Some common services owned by the operating system include process scheduling management, resource management, memory management, file systems, network management, security management, storage management, and input-output device management. In this research, the discussion will be pursed into one of the Operating System service that is process scheduling management. One of the derivative material of process scheduling that is difficult for students to understand is Concurrency. In general, concurrency is the processes (more than one process) that occur at the same time. Some of the algorithms used in the application of concurrency in process management include Dining Philosophers, Banker Algorithm, Producer-Consumer, and Readers-Writers. Explanations related to these four algorithms require logic and a high degree of seriousness. Unfortunately not all students are able to maintain their concentration in a long time to learn the logic flow of the four algorithms. Based on the above description, the authors want to design a software that is able to simulate the four algorithms into the form of multimedia-based learning media that is expected to increase the attractiveness and capture power of students who study it.*
Keywords : *Simulation, Concurrency, Operating System, Multimedia*

Abstrak : Mata kuliah sistem operasi merupakan salah satu mata kuliah wajib yang harus diajarkan kepada mahasiswa yang mengambil jurusan / program studi teknik informatika / sistem informasi / teknologi informasi. Mata kuliah ini memperkenalkan kepada mahasiswa mengenai konsep sistem operasi, jenis-jenis sistem operasi yang digunakan dalam kehidupan sehari-hari, layanan-layanan umum yang dimiliki oleh sistem operasi, dan mempelajari teknik dan algoritma yang digunakan dalam penerapan / pengelolaan layanan di sistem operasi. Sistem operasi adalah suatu program yang bertindak sebagai perantara antara user, program aplikasi dan perangkat keras. Beberapa layanan umum yang dimiliki oleh sistem operasi antara lain manajemen penjadwalan proses, manajemen sumber daya (*resource*), manajemen memori, sistem berkas, manajenen jaringan, manajemen keamanan, manajemen penyimpanan, dan manajemen perangkat masukan-keluaran. Dalam penelitian kali ini, pembahasan akan dikerucutkan ke dalam salah satu layanan Sistem Operasi yaitu manajemen penjadwalan proses. Salah satu materi turunan dari manajemen proses yang sulit untuk dipahami oleh mahasiswa adalah *Concurrency*. Secara umum *concurrency* merupakan proses-proses (lebih dari satu proses) yang terjadi pada saat bersamaan. Beberapa algoritma yang digunakan dalam penerapan *concurrency* pada manajemen proses antara lain *Dining Philosophers, Banker Algorithm, Producer-Consumer*, dan *Readers-Writers*. Penjelasan terkait keempat algoritma tersebut membutuhkan logika dan tingkat keseriusan yang tinggi. Sayangnya tidak semua mahasiswa mampu untuk mempertahankan konsentrasinya dalam waktu yang lama untuk mempelajari alur logika dari keempat algoritma tersebut. Berdasarkan uraian di atas, penulis ingin merancang suatu perangkat lunak yang mampu untuk mensimulasikan keempat algoritma tersebut ke dalam bentuk media pembelajaran berbasis multimedia sehingga diharapkan akan meningkatkan daya tarik dan daya tangkap dari mahasiswa yang mempelajarinya.
Kata Kunci : Simulasi, *Concurrency,* Sistem Operasi, Multimedia

I. PENDAHULUAN

Sistem operasi adalah suatu program yang bertindak sebagai perantara antara user, program aplikasi dan perangkat keras. Mata kuliah sistem operasi merupakan salah satu mata kuliah wajib yang harus diajarkan kepada mahasiswa yang mengambil jurusan / program studi teknik informatika / sistem informasi / teknologi informasi. Mata kuliah ini memperkenalkan kepada mahasiswa mengenai konsep sistem operasi, jenis-jenis sistem operasi yang digunakan dalam kehidupan sehari-hari, layanan-layanan umum yang dimiliki oleh sistem operasi, dan mempelajari teknik dan algoritma yang digunakan dalam penerapan / pengelolaan layanan di sistem operasi.

Salah satu layanan yang dimiliki oleh sistem operasi yaitu manajemen penjadwalan proses dan manajemen sumber daya (*resource*). Dalam penelitian kali ini,

pembahasan akan fokus kepada materi *concurrency*, salah satu layanan Sistem Operasi yang merupakan materi turunan dari manajemen proses yang sulit untuk dipahami oleh mahasiswa. Secara umum *concurrency* merupakan proses-proses (lebih dari satu proses) yang terjadi pada saat bersamaan. Beberapa algoritma yang digunakan dalam penerapan *concurrency* pada manajemen proses antara lain *Dining Philosophers* (mewakili *mutual exclusion*), *Banker Algorithm* (mewakili *deadlock*), *Producer-Consumer* (mewakili *bounded buffer*), dan *Readers-Writers* (mewakili sinkronisasi / *starvation*).

Penjelasan terkait keempat algoritma tersebut membutuhkan logika dan tingkat keseriusan yang tinggi. Sayangnya tidak semua mahasiswa mampu untuk mempertahankan konsentrasinya dalam waktu yang lama untuk mempelajari alur logika dari keempat algoritma tersebut.

Berdasarkan uraian di atas, penulis ingin merancang suatu perangkat lunak yang mampu untuk mensimulasikan keempat algoritma tersebut ke dalam bentuk media pembelajaran berbasis multimedia sehingga diharapkan akan meningkatkan daya tarik dan daya tangkap dari mahasiswa yang mempelajarinya.

II. KAJIAN PUSTAKA

II.1 Simulasi

Pembuatan simulasi tidak lepas dengan bantuan program komputer untuk mewujudkannya. Simulasi komputer adalah program komputer yang berfungsi untuk menirukan perilaku sistem nyata tertentu, memiliki sifat *Physical* and *Interactive*. Simulasi adalah penggambaran suatu sistem atau proses dengan peragaan memakai model statistik atau pemeranan (Maulani, 2017). Simulasi merupakan pemodelan dari situasi *real life* ke dalam komputer sehingga dapat dipelajari bagaimana cara kerjanya (Yulianto & Haryana, 2016). Definisi lain menurut Law dan Kelton (1991), simulasi merupakan sekumpulan metode dan aplikasi untuk menirukan atau mereprestasikan perilaku dari suatu sistem nyata, yang biasanya dilakukan pada komputer dengan menggunakan perangkat lunak tertentu.

Contoh dari simulasi komputer antara lain *training simulation, medical simulator, city simulator (sim city), flight simulator, simulation game, engineering simulation*, simulasi antrian layanan bank, simulasi perang, *power plan simulation,* dan lain-lain. Tahapan dalam mengembangkan simulasi komputer adalah (Febrian, 2002) :

1. Memahami sistem yang akan disimulasikan.
2. Mengembangkan model matematika dari sistem.
3. Mengembangkan model matematika untuk simulasi.
4. Membuat program (*software*) komputer.
5. Menguji, memverifikasi dan memvalidasi keluaran simulasi.

II.2 *Concurrency*

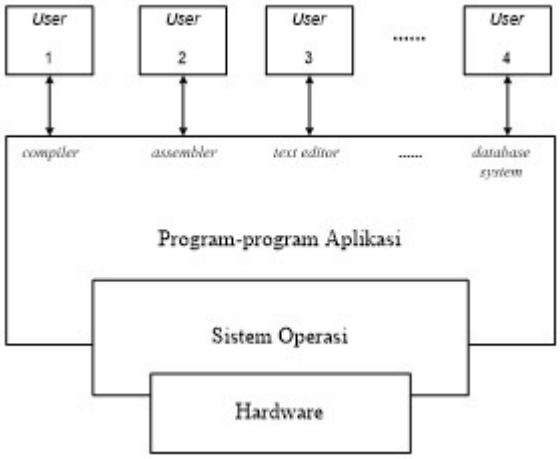
Konkurensi merupakan salah satu kejadian (*event*) di dalam manajemen proses, dimana proses-proses (lebih dari satu proses) yang terjadi pada saat bersamaan. Pada proses-proses konkuren yang berinteraksi mempunyai beberapa masalah yang harus diselesaikan, antara lain (Tandri, 2006) :

1. *Mutual Exclusion,* adalah jaminan hanya satu proses yang mengakses sumber daya pada satu interval waktu tertentu. *Mutual Exclusion* merupakan kondisi dimana terdapat sumber daya yang tidak dapat dipakai bersama pada waktu yang bersamaan (misalnya : printer, disk drive). Kondisi demikian disebut sumber daya kritis, dan bagian program yang menggunakan sumber daya kritis disebut *critical region / section.*
2. Sinkronisasi, adalah proses pengaturan jalannya beberapa proses pada saat yang bersamaan. Tujuan utama sinkronisasi adalah menghindari terjadinya inkonsitensi data karena pengaksesan oleh beberapa proses yang berbeda (*mutual exclusion*) serta untuk mengatur urutan jalannya proses-proses sehingga dapat berjalan dengan lancar dan terhindar dari *race condition, deadlock* dan *starvation*
3. *Deadlock,* suatu *event* dimana proses menunggu satu kejadian tertentu yang tak akan pernah terjadi. Sekumpulan proses berkondisi *deadlock* bila setiap proses yang ada di kumpulan itu menunggu suatu kejadian yang hanya dapat dilakukan proses lain yang juga berada di kumpulan itu. Proses menunggu kejadian yang tidak akan pernah terjadi.
4. *Starvation,* suatu *event* dimana proses kekurangan *resource* (karena terjadi *deadlock*) dan tidak pernah mendapat *resource* yang dibutuhkan sehingga mengalami *starvation* (kelaparan). *Starvation* juga bisa terjadi tanpa *deadlock* ketika terdapat kesalahan dalam sistem sehingga terjadi ketimpangan dalam pembagian *resouce*. Satu proses selalu mendapat *resource*, sedangkan proses yang lain tidak pernah mendapatkannya.

II.3 Sistem Operasi

Secara umum, sebuah sistem komputer terbagi atas hardware, sistem operasi, program aplikasi, dan user. Komponen-komponen sistem komputer tersebut ditunjukkan oleh gambar 1 berikut ini (Canharta, 2006).

p-ISSN : 2460-7363
e-ISSN : 2614-6606



Gambar 1. Komponen-Komponen Sistem Komputer

Menurut Silberschatz, Galvin, dan Gagne (2007), pada umumnya sebuah sistem operasi modern mempunyai layanan-layanan sebagai berikut:

1. Manajemen Proses, proses adalah program yang sedang di eksekusi. Suatu proses membutuhkan satu atau beberapa sumber daya untuk menyelesaikan tugasnya. Sumber daya tersebut dapat berupa *CPU time*, memori, berkas-berkas, atau perangkat-perangkat I/O.
2. Manajemen Memori, memori adalah suatu array besar yang terdiri dari word atau *byte*, yang ukurannya dapat mencapai ratusan, ribuan, atau bahkan jutaan. Setiap word atau byte mempunyai alamat tersendiri. Memori berfungsi sebagai tempat penyimpanan data yang digunakan oleh CPU atau perangkat I/O. Memori merupakan tempat penyimpanan data yang sementara (*volatile*), artinya data dapat hilang begitu sistem dimatikan.
3. Manajemen *Secondary Storage*, data yang disimpan dalam memori bersifat sementara dan ukurannya sangat kecil jika dibandingkan dengan keseluruhan data yang terdapat dalam komputer. Oleh karena itu, untuk meyimpan keseluruhan data dan program komputer dibutuhkan *secondary storage* yang bersifat *non volatile* dan mampu menampung banyak data. Contoh dari secondary storage adalah harddisk, disket, dan lain-lain.
4. Manajemen Sistem I/O, fungsi ini sering disebut *device manager* dimana sistem operasi menyediakan *device driver* yang umum sehingga operasi I/O dapat seragam membaca atau menuliskan data tanpa mempedulikan mekanisme kerja yang berbeda dari perangkat-perangkat I/O yang ada.
5. Manajemen Arsip, *file* adalah kumpulan informasi yang dibuat dengan tujuan tertentu. *File* disimpan dalam struktur yang bersifat hirarkis, seperti direktori.
6. Sistem Proteksi, proteksi mengacu pada mekanisme untuk mengontrol akses yang dilakukan oleh program, prosesor, atau pengguna ke sistem sumber daya.
7. Sistem Jaringan, mendukung penggunaan jaringan. Sistem ini umumnya kini telah terpadu dalam sistem operasi karena kebutuhan kinerjanya serta kebutuhan komputasi telah menghendaki kemampuan ini
8. *User Interface (Shell)*, Sistem Operasi menunggu instruksi dari pengguna (*command driven*). Program yang membaca instruksi dan mengartikan *control statements* disebut *control-card interpreter* atau *command-line interpreter*. *User Interface* sangat bervariasi dari satu sistem operasi ke sistem operasi yang lain dan disesuaikan dengan tujuan dan teknologi *I/O devices* yang ada. Contohnya: *Command Line Interface (CLI), Graphical User Interface (GUI),* dan lain-lain.

II.4 Multimedia

Multimedia yang berasal dari kata multi yang berarti banyak atau lebih dari satu dan media yang dapat diartikan penyajian suatu tempat. Multimedia adalah pemanfaatan komputer untuk membuat dan menggabungkan teks, audio, gambar, bergerak (video dan animasi dengan menggunakan *link* dan *tool* yang memungkinkan pemakai (*user*) melakukan navigasi, berinteraksi, berkreasi dan berkomunikasi (Suyanto, 2004).

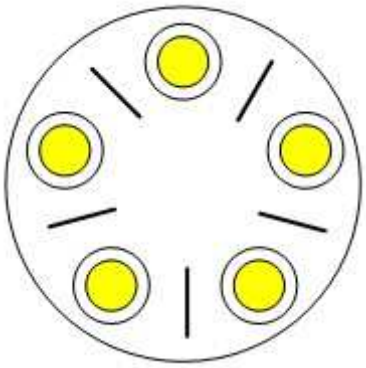
Definisi lain dari multimedia menurut Hofstetter (2001) adalah penggunaan komputer untuk menampilkan informasi yang merupakan gabungan dari teks, grafik, audio dan video sehingga membuat pengguna dapat bernavigasi, berinteraksi, berkreasi dan berkomunikasi dengan komputer. Selain kombinasi dari objek-objek multimedia tersebut, terdapat juga 4 komponen penting lainnya, yaitu:

1. Adanya komputer untuk mengatur apa yang akan dilihat dan didengar, dan apa yang akan berinteraksi dengan penggunanya
2. Adanya *link-link* yang menghubungkan informasi-informasi yang tersedia
3. Adanya *tool-tool* navigasi bagi pengguna agar dapat menggunakan informasi yang tersedia
4. Adanya prosedur bagi pengguna untuk mengumpulkan, memproses dan menyampaikan informasi dan ide-idenya.

Kelebihan multimedia adalah menarik indera dan minat, karena merupakan gabungan antara pandangan, suara dan gerakan. Lembaga Riset dan Penerbitan Komputer yaitu *Computer Technology Research (CTR)*, menyatakan bahwa orang hanya mampu mengingat 20% dari yang dilihat dan 30% dari yang didengar, tetapi orang dapat mengingat 50% dari yang dilihat dan didengar dan 30% dari yang dilihat, didengar dan dilakukan sekaligus. Multimedia menjadi *tool* yang ampuh untuk pengajaran dan pendidikan (Suyanto, 2004).

II.5 Dining Philosophers Problem

Dining Philosophers Problem merupakan salah satu masalah klasik dalam *mutual exclusion. Dining Philosohers Problem* dapat diilustrasikan sebagaimana gambar 2 berikut, terdapat lima orang filsuf yang sedang duduk mengelilingi sebuah meja. Terdapat lima mangkuk mie di depan masing-masing filsuf dan satu sumpit di antara masing-masing filsuf. Para filsuf menghabiskan waktu dengan berpikir (ketika kenyang) dan makan (ketika lapar). Ketika lapar, filsuf akan mengambil dua buah sumpit (di tangan kiri dan tangan kanan) lalu makan, namun adakalanya, hanya terdapat satu sumpit saja. Jika ada filsuf yang mengambil dua buah sumpit, maka dua filsuf di samping filsuf yang sedang makan harus menunggu sampai sumpit ditaruh kembali. Hal ini dapat diimplementasikan dengan *wait* dan *signal* (Canharta, 2006).



Gambar 2. *Dining Philosohers Problem*

Struktur proses atau dikenal dengan spesifikasi proses (PSPEC) dari *Dining Philosohers Problem* dapat dilihat pada algoritma di bawah ini :

```
repeat
      wait (sumpit[i]);
      wait (sumpit [i+1 mod 5]);
            ...
            makan
            ...
      signal (sumpit[i]);
      signal (sumpit[i+1 mod 5]);
            ...
            berpikir
            ...
until false
```

Meskipun solusi ini menjamin bahwa tidak ada 2 filsuf yang makan bersama-sama, namun masih mungkin terjadi *deadlock*, yaitu jika tiap-tiap filsuf lapar dan mengambil sumpit kiri, maka semua nilai sumpit=0, dan kemudian tiap-tiap filsuf akan mengambil sumpit kanan, maka akan terjadi deadlock.

II.6 Banker Algorithm Problem

Algoritma *Banker* dikemukakan oleh Edsger W.Dijkstra dan merupakan sebuah strategi untuk menghindari *deadlock.* Algoritma ini disebut algoritma *Banker* karena memodelkan *banker* di kota kecil yang berurusan dengan sekumpulan nasabah yang memohon kredit. Algoritma ini mencegah terjadinya *deadlock* dengan memutuskan apakah menyetujui atau menunda permintaan sumber daya oleh proses. Ketika sebuah proses meminta sumber daya, maka permintaan tersebut harus diperiksa oleh bankir.

Analogi dari algoritma *Banker* dengan sistem operasi adalah :

1. Nasabah, merupakan proses-proses yang sedang berjalan. Dalam algoritma *Banker*, setiap nasabah memiliki batas kredit dan apabila seorang nasabah telah mencapai batas kredit pinjaman maksimum, maka diasumsikan nasabah tersebut telah menyelesaikan semua permasalahan bisnisnya dan dapat mengembalikan semua pinjamannya kepada bank.
2. Bankir, merupakan sistem operasi. Bankir adalah seorang peminjam yang konservatif. Ketika sebuah proses (nasabah) meminta peminjaman sumber daya (uang), bankir melihat buku bank dengan cermat dan berusaha untuk memutuskan apakah peminjaman tersebut dapat menyebabkan keadaan *deadlock* (setelah peminjaman tersebut disetujui) atau tidak.
3. Uang, yaitu dana yang dimiliki bank merupakan sumber daya. Setelah peminjaman, akan terdapat sejumlah sumber daya tersisa di dalam sistem. Kita asumsikan masing-masing proses meminta batas maksimum sumber daya. Jika bankir memiliki cukup sumber daya yang tersisa untuk memastikan bahwa semua proses dapat berakhir

dengan aman. Ini menunjukkan keadaan berada dalam *safe state* dan peminjaman disetujui.

Kelemahan dari algoritma *Banker* adalah setelah peminjaman, bankir tidak dapat menjamin semua proses dapat berakhir dengan semestinya, sehingga terjadi *unsafe state*. Dalam kasus ini, peminjaman akan ditunda atau diblok hingga peminjaman tidak menyebabkan *unsafe state* pada sistem. Pada algoritma *Banker* ini, kondisi *mutual exclusion, hold-and-wait*, dan *no-preemption* diijinkan dan proses-proses melakukan klaim penggunaan sumber daya yang diperlukan. Proses-proses diizinkan mengenggam sumber daya-sumber daya sambil meminta dan menunggu sumber daya lain, sementara sumber daya yang sedang digunakan itu tidak diijinkan untuk di-*preempt* oleh proses lain (Zhuang, 2006).

II.7 Producer-Consumer Problem

Kasus *producer-consumer* digunakan sebagai ilustrasi pembahasan sinkronisasi. Masalah *producer-consumer* disebut juga *bounded-buffer* problem (masalah *buffer* dengan jumlah terbatas). Asumsi yang digunakan dalam *producer-consumer problem* adalah sebagai berikut (Budiman, 2006):

1. Dua proses menggunakan suatu *buffer* yang dipakai bersama dan berukuran tetap.
2. Satu proses adalah *producer* yang meletakkan informasi ke *buffer*.
3. Proses lain adalah *consumer* yang mengambil informasi dari *buffer*.

Masalah *producer-consumer* dapat dikembangkan menjadi masalah yang memiliki m buah produsen dan n buah konsumen. Selanjutnya, dikarena *buffer* terbatas, masalah berikut dapat terjadi, yaitu:

1. Masalah untuk *producer*, terjadi ketika *buffer* telah penuh, sementara *producer* ingin meletakkan informasi ke *buffer* yang telah penuh itu.
2. Masalah untuk *consumer*, terjadi ketika *consumer* ingin mengambil informasi sementara *buffer* telah/sedang kosong.

Struktur proses pada *producers-consumers problem* dapat dilihat pada PSPEC di bawah ini.

```
producer:
while (true) {
   /* produce item v */
   b[in] = v;
   in++;
}
```

```
consumer:
while (true) {
   while (in <= out)
      /*do  nothing */;
   w = b[out];
   out++;
   /* consume item w */
}
```

Kedua proses memerlukan sinkronisasi agar keduanya dapat menghindari masalah. Penyelesaian ini memiliki dua rutin, yaitu *sleep* dan *Wake-up*. Kedua rutin bersifat atomik, yaitu saat rutin dieksekusi maka tak ada interupsi yang dapat menyela (interupsi). *Sleep* adalah rutin yang menyebabkan pemanggil di-*blocked*, ditunda sampai proses lain membangunkan (*Wake-up*). *Wake-up* adalah rutin untuk membangunkan proses yang sedang berada dalam status *sleeping*. *Wake-up* mempunyai satu parameter, yaitu proses yang dibangunkan.

Solusi penyelesaian dengan *sleep and wakeup* adalah sebagai berikut (Budiman, 2006):

1. Solusi untuk masalah *producer*. *Producer* memanggil *sleep* setelah mengetahui *buffer* telah penuh saat *producer* akan menyimpan informasi ke *buffer*. *Producer* tidak lagi aktif kecuali dibangunkan (*wake-up*). Proses lain (*consumer*) akan memberitahu bahwa satu item atau lebih telah diambil dari *buffer* sehingga terdapat ruang bagi *producer* untuk menyimpan informasi kembali ke *buffer*.
2. Solusi untuk masalah *consumer*. *Consumer* memanggil *sleep* begitu mengetahui buffer telah kosong saat *consumer* mengambil item. *Consumer* tidak lagi aktif kecuali dibangunkan (*wake-up*) oleh proses lain (*producer*) yang memberitahu bahwa *buffer* telah terisi satu item atau lebih sehingga terdapat informasi yang dapat diambil *consumer* dari *buffer*.

II.8 Readers and Writers Problem

Masalah pembaca dan penulis (*readers and writers problem*) memodelkan pengaksesan lebih dari satu proses ke basis data yang sama. Masalahnya dapat dideskripsikan sebagai berikut, diasumsikan terdapat basis data besar seperti sistem reservasi penerbangan dengan proses-proses yang berkompetisi untuk membaca dan menulis pada basis data tersebut. Diasumsikan bahwa pada sistem basis data tersebut memiliki prosedur sebagai berikut (Tandri, 2006) :

1. Sistem reservasi mengijinkan banyak proses membaca (*reader*) basis data pada saat yang sama.
2. Jika terdapat satu proses menulis / mengubah (*writer*) basis data, proses lain tidak boleh mengakses basis data baik membaca atau menulis.

Pada masalah ini, *writers* memiliki prioritas yang lebih tinggi daripada *readers*. Jika ada *writer* yang sedang menunggu, maka tidak boleh ada *reader* lain yang bekerja. *Writer* akan memblok semua proses *reading* dan melakukan proses *writing*. Ketika proses *writing* selesai, maka proses *reading* dapat dilanjutkan kembali oleh *readers*. Struktur proses pada *readers and writers problem* dapat dilihat pada PSPEC di bawah ini.

p-ISSN : 2460-7363
e-ISSN : 2614-6606

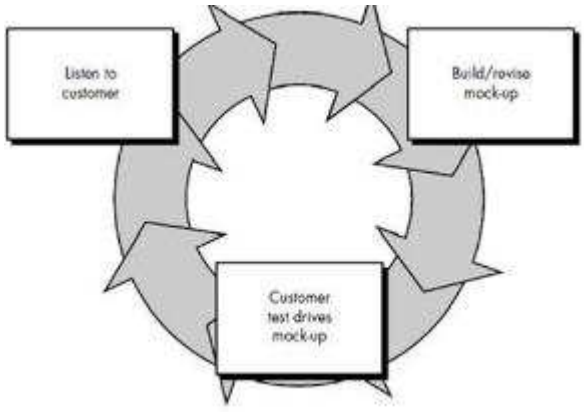
```
// Struktur Proses Writer
      wait(wrt);
            ...
            menulis
            ...
      signal(wrt);

// Struktur Proses Reader
      wait(mutex);
            readcount:= readcount + 1;
            if readcount = 1 then wait(wrt);
      signal(mutex);
                  ...
            membaca
                  ...
      wait(mutex);
            readcount:= readcount - 1;
            if readcount = 0 then signal(wrt);
      signal(mutex);
```

Algoritma di atas menunjukkan masalah *reader* dan *writer*. Jika ada *writer* dalam *critical section* dan ada n *reader* yang menunggu, maka satu *reader* akan antri di wrt, dan n – 1 *reader* akan antri. Jika *writer* mengeksekusi signal(wrt), maka dapat disimpulkan bahwa eksekusi adalah menunggu *reader* atau menunggu satu *writer*.

III. METODE PENELITIAN

Metode penelitian yang digunakan oleh penulis adalah metode deskriptif, yaitu penelitian yang dilakukan untuk mengetahui nilai variabel mandiri, baik satu variabel atau lebih (independen) tanpa membuat perbandingan, atau menghubungkan dengan variabel yang lain (Sugiyono, 2003). Teknik pengumpulan data yang digunakan teknik wawancara.

Adapun metode pengembangan sistem yang digunakan yaitu memakai tahapan siklus hidup perangkat lunak / *software development life cycle (SDLC)* dengan model *prototyping*. Adapun tahapan-tahapan aktivitas dari model *prototyping* dapat dilihat pada gambar 3 di bawah ini.



Gambar 2. Model *Prototyping*

Tahapan dari Model *Prototyping* merefleksikan pokok-pokok dari aktivitas pengembangan perangkat lunak sebagai berikut:

1. *Listen to Customer*

   Pada tahap *listen to customer*, kegiatan penelitian dibagi menjadi 2 fase yaitu :

   a. Persiapan Penelitian

      Pada tahap ini penulis melakukan wawancara dengan mahasiswa dalam mengumpulkan data. Data yang menjadi fokus utama dalam penelitian ini yaitu bagaimana tingkat pemahaman mahasiswa terhadap salah satu silabus pada mata kuliah Sistem Operasi yaitu Konkurensi baik dari sisi konsep, alur proses, maupun algoritmanya.

   b. Analisis

      Pada tahap ini dilakukan identifikasi tingkat pemahaman mahasiswa terhadap mata kuliah Sistem Operasi berdasarkan hasil pengumpulan data nilai UTS. Data primer yang dibutuhkan adalah data nilai UTS selama 2 tahun terakhir.

2. *Build/Revise Mock Up*

   Tahap ini merupakan desain sistem/perancangan model dan alur pembangunan simulasi pembelajaran konkurensi berbasis multimedia. Perancangan sistem yang dibuat meliputi pembuatan *storyboard*, tahapan proses, algoritma, simulasi tabel, dan *sitemap design.*

3. *Customer Test Drives Mock-Up*

   Setelah tahap *build/revise mock-up* selesai, maka tahap terakhir yaitu *customer test drives mock-up*/pengujian *prototype* serta hasil analisis dari pengujian. Tahap ini merupakan tahap utama dari sistem yang dibuat karena pada tahapan ini dilakukan *user acceptable test* yang dilanjutkan dengan pengujian soal dan pengisian kuisioner untuk mengetahui pemahaman mahasiswa terkait sebelum dan setelah berinteraksi dengan simulasi pembelajaran konkurensi.

IV. PEMBAHASAN

IV.1 Spesifikasi Proses (PSPEK)

Simulasi Pembelajaran *Concurrency* Pada Mata Kuliah Sistem Operasi Berbasis Multimedia memiliki 4 (empat) buah modul pembelajaran utama, yaitu *Dining Philosophers Problem, Banker Algorithm Problem, Producer-Consumer Problem,* dan *Readers-Writers Problem.*

Spesifikasi proses dari masing-masing modul dapat kita bahas sebagai berikut :

1. *Dining Philosophers Problem*
   a. Jumlah filsuf : 5 (lima) orang
   b. Jumlah sumpit : 5 (lima) buah
   c. Kondisi status dari masing-masing filsuf :
      i. Kenyang → Berpikir (detik)

ii. Lapar → Makan (detik)
iii. Kelaparan → Mati (detik)

d. Storyboard :

Di dalam perangkat lunak, kondisi filsuf di atas dirancang penulis secara matematis menjadi waktu-A dan waktu-B.

i. Waktu-A, yaitu waktu yang diperlukan untuk mengubah kondisi filsuf dari kenyang menjadi lapar (dalam sekon).
ii. Waktu-B, yaitu waktu yang diperlukan untuk mengubah kondisi filsuf dari lapar menjadi mati (dalam sekon).
iii. Masa Hidup, yaitu masa hidup filsuf pada saat itu (dalam sekon). Maksimum masa hidup adalah sebesar nilai waktu-A + waktu-B. Masa hidup filsuf akan bertambah ketika filsuf sedang makan dan akan berkurang di luar status itu.

2. *Banker Algorithm Problem*

a. Objek yang terlibat :
i. *Resource* : jumlah uang yang dimiliki oleh bank
ii. *Customer* : batas maksimum pemberian kredit
iii. *Bankir* : pemutus kredit
iv. Kasir : perantara antara *bankir* dan *customer*

b. Storyboard :
i. Ketika *customer* memohon kredit, maka permohonan kredit tersebut harus diperiksa oleh *bankir*.
ii. Dalam algoritma *Banker*, setiap *customer* memiliki batas maksimum kredit dan apabila seorang *customer* telah mencapai batas maksimum kredit, maka diasumsikan *customer* tersebut tidak dapat mendapatkan pinjaman dan hanya dapat mengembalikan semua pinjamannya kepada bank secara bertahap. Ada kalanya, seorang customer dapat menyelesaikan semua permasalahan bisnisnya dan mengembalikan semua pinjamannya kepada bank sebelum mencapai batas kredit maksimum.

c. Algoritma *Banker* terdiri atas algoritma *Safety* dan algoritma *Resource Request*. Cara kerja dari masing-masing algoritma tersebut adalah sebagai berikut:
i. Algoritma *Safety*
a) Set nilai *Work* = *Available* (*Available* = dana masih tersisa setelah peminjaman).
b) Set nilai *Finish*[i] = *False* untuk semua *customer*.
c) Cari semua *customer* yang memiliki *Needs* (keperluan) lebih kecil atau sama dengan nilai *Work*. Besar *Needs* dihitung dengan mengurangkan nilai *maksimum_resource_customer* dengan nilai *allocation_resource_customer*).
d) Apabila tidak ada lagi *customer* dengan status *Needs* <= *Work*, maka periksa apakah semua Finish[i] bernilai *True*. Apabila semua Finish[i] = *True*, maka kondisi setelah peminjaman adalah *safe*. Apabila tidak, maka kondisi setelah peminjaman adalah *unsafe*.
ii. Algoritma *Resource Request*
a) Jika permohonan kredit (*Request*) lebih besar dari sisa *resource* pada bank (*Available*), maka permohonan kredit ditunda (*pending*).
b) Bankir melakukan simulasi peminjaman *resource* kepada *customer* dimana *available resource* pada bank dikurangi dengan besar *request*.
c) Jalankan algoritma *Safety*.
d) Apabila algoritma *Safety* menghasilkan keadaan *safe*, maka permohonan kredit direalisasikan. Apabila algoritma *Safety* menghasilkan keadaan *unsafe*, maka permohonan kredit ditunda (*pending*).

3. *Producer-Consumer Problem*

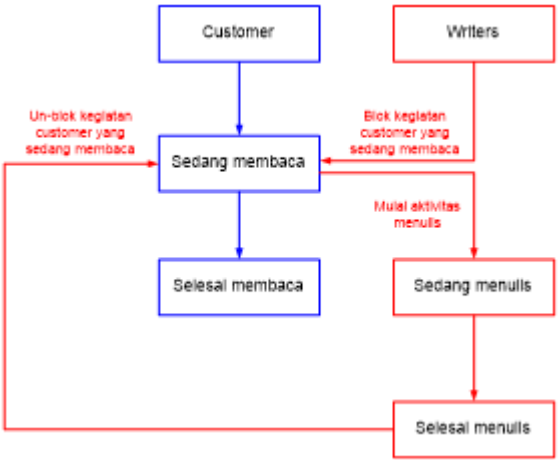
a. Objek yang terlibat : *producer, consumer, market (buffer)*
b. Inisialisasi variabel :
i. Pengaturan pada *Producer*, yaitu jumlah *producer*, batas maksimum dan minimum bagi *producer* dalam satu kali produksi.
ii. Pengaturan pada *Consumer*, yaitu jumlah *consumer*, batas maksimum dan minimum bagi *consumer* dalam satu kali konsumsi.
iii. Pengaturan pada *Market*, yaitu batas ukuran maksimum dan minimum *market*.

c. Storyboard :

p-ISSN : 2460-7363
e-ISSN : 2614-6606

i. *Producer* aktif memproduksi dan meletakkan item ke *market* (*buffer*). Aksi ini akan menambah jumlah item di dalam *market*.
ii. *Consumer* aktif mengambil item dari *market (buffer)* dan mengonsumsi item. Aksi ini akan mengurangi jumlah item di dalam *market*.
iii. Apabila *market* telah penuh atau jumlah item di dalam *market* telah mencapai batas maksimum, maka *producer* akan tidur (memanggil aksi *sleep*).
iv. Apabila *consumer* mengambil item dari *market* dan *producer* dalam keadaan *sleep*, maka *consumer* akan membangunkan (*wake-up*) *producer*.
v. Apabila *market* kosong atau jumlah item di dalam *market* telah mencapai batas minimum, maka *consumer* akan tidur (memanggil aksi *sleep*).
vi. Apabila *producer* meletakkan item ke *market* dan *consumer* dalam keadaan *sleep*, maka *producer* akan membangunkan (*wake-up*) consumer.

4. *Readers-Writers Problem*
   a. Objek yang terlibat : *readers, writers, database*
   b. *Storyboard* : *writers* memiliki prioritas lebih tinggi dari *readers*. *Writers* mempunyai hak menulis basis data kapan saja. Ketika *writers* ingin menulis dan *readers* sedang membaca basis data, maka *readers* akan diblok dan tidak boleh membaca basis data hingga *writers* selesai menulis.
   c. Diagram proses dari *Readers-Writers Problem* dapat dilihat pada gambar 3 berikut.



Gambar 3. Diagram Proses *Readers-Writers Problem*

IV.2 Tabel Simulasi

Pemodelan terhadap Simulasi Pembelajaran *Concurrency* Pada Mata Kuliah Sistem Operasi Berbasis Multimedia akan lebih mudah apabila digambarkan dalam bentuk tabel simulasi. Kita akan memakai studi kasus dari masing-masing modul untuk dibuat tabel simulasi yang akan dijelaskan sebagai berikut :

1. *Dining Philosophers Problem*

   Langkah awal yang akan dilakukan adalah memberikan inisialisasi variabel awal sebagai berikut :
   a. input PHILOSOPHERS = 5
   b. array of CHOPSTICKS (1,2,3,4,5)
   c. array of PHILOSOPHERS_STATUS [Think, Eat, Hungry] = (5,2,10); (4,4,10); (3,6,10); (8,1,10); (6,3,10)
   d. array of PHILOSOPHERS_STICKS [*Chopsticks1, Chopsticks2*] = (5,1); (1,2); (2,3); (3,4); (4,5)

   maka tabel simulasi yang dapat dibuat dapat dilihat pada tabel 1 berikut

Tabel 1. Simulasi *Dining Philosophers Problem*

	1	2	3	4	5	6	7	8	9	10	11	12
P1	T	T	T	T	T	E	E	T	T	T	T	T
P2	T	T	T	T	H	H	H	H	H	E	E	E
P3	T	T	T	E	E	E	E	E	E	T	T	T
P4	T	T	T	T	T	T	T	T	H	H	E	T
P5	T	T	T	T	T	T	H	E	E	E	T	T

2. *Banker Algorithm Problem*

   Inisialisasi variabel awal *banker algorithm problem* dapat dideklarasikan sebagai berikut :
   a. input NUMBER_OF_PROCESS = 5
   b. input NUMBER_OF_RESOURCES = 3
   c. array of RESOURCES [index,instances] = (1,10); (2,5); (3,7)
   d. array of PROCESS_RES_ALLOCATION = [P, R1, R2, R3] = (1,0,1,0); (2,2,0,0); (3,3,0,2); (4,2,1,1); (5,0,0,2)
   e. array of PROCESS_RES_MAX = [P, R1, R2, R3] = (1,7,5,3); (2,3,2,2); (3,9,0,2); (4,2,2,2); (5,4,3,3)

   maka tabel simulasi yang dapat dibuat dapat dilihat pada tabel 2 berikut

Tabel 2. Simulasi *Banker Algorithm Problem*

Process	Allocation	Max	Need	Available
	A B C	A B C	A B C	A B C
P0	0 1 0	7 5 3	7 4 3	
P1	2 0 0	3 2 2	1 2 2	
P2	3 0 2	9 0 2	6 0 0	3 3 2
P3	2 1 1	2 2 2	0 1 1	
P4	0 0 2	4 3 3	4 3 1	

3. *Producer-Consumer Problem*

   Dengan memberikan inisialisasi variabel awal sebagai berikut :
   e. int BUFFER_SIZE = 0

p-ISSN : 2460-7363
e-ISSN : 2614-6606

f. int COUNTER = 0
g. int ITEM = 0
h. boolean PRODUCER = False
i. boolean CONSUMER = False
j. input BUFFER_SIZE = 10
k. input COUNTER = 0
l. array of PRODUCER (2,0,5,8,4,0,6,0)
m. array of CONSUMER (0,3,0,0,0,6,0,3)

maka tabel simulasi yang dapat dibuat dapat dilihat pada tabel 3 berikut

Tabel 3. Simulasi *Producer-Consumer Problem*

M / C	Item	Counter	Relation	Buffer_Size	M	C
-	0	0	<=	10	T	F
M	2	2	<=	10	T	T
C	3	0	<=	10	T	F
M	5	5	<=	10	T	T
M	8	10	>=	10	F	T
M	4	*End*	*End*	10	*End*	*End*
C	6	4	<=	10	T	T
M	6	10	=	10	F	T
C	3	7	<=	10	T	T

4. *Readers-Writers Problem*

Inisialisasi variabel awal dari *readers-writers problem* adalah sebagai berikut :

a. boolean ACTIVE=False
b. boolean WAIT=False
c. array of READERS [(2,2);(3,4);(5,3);(8,4);(9,4);(12,3);(16,3)]
d. array of WRITERS [(6,5);(10,6);(14,4)]

maka tabel simulasi yang dapat dibuat dapat dilihat pada tabel 4 berikut

Tabel 4. Simulasi *Readers-Writers Problem*

0	1	2	3	4	5	6	7	8	9	10	11	12
		AR	AR		AR	AR	AR	WR	WR	WR	WR	WR
			AR	AR	AR	AR			WR	WR	WR	WR
						WW	WW	AW	AW	AW	AW	AW
										WW	WW	WW
												WR

13	14	15	16	17	18	19	20	21	22	23	24	25
AR	AR	AR	AR									
AR	AR	AR	AR									
	WW	WW	WW	WW	WW	WW	WW	WW	WW	WW	WW	WW
WW	WW	WW	WW	AW	AW	AW	AW	AW	AW			
WR	WR	WR	WR	WR	WR	WR	WR	WR	WR	AR	AR	AR
			AR	AR	AR							

V. IMPLEMENTASI

Simulasi Pembelajaran *Concurrency* Pada Mata Kuliah Sistem Operasi Berbasis Multimedia yang dibangun berfungsi sebagai suplemen bahan ajar pada mata kuliah Sistem Operasi.

Tampilan antar muka dari fitur-fitur utama dari masing-masing modul dapat dilihat sebagai berikut :

1. Menu Utama simulasi Pembelajaran *Concurrency*, merupakan portal utama dari masing-masing modul dapat dilihat pada gambar 4 di bawah ini.

Gambar 4. Portal Utama (*Dashboard)*

2. *Dining Philosophers Problem*, fungsional yang akan ditampilkan antara lain :

a. Inisialisasi simulasi, dapat dilihat pada gambar 5 berikut.

Gambar 5. Inisialisasi Simulasi *Dining Philosophers Problem*

b. Kondisi awal semua filsuf sedang berpikir dalam kondisi kenyang, dapat dilihat pada gambar 6 berikut.

Gambar 6. Simulasi Kondisi Filsuf Berpikir Dalam Keadaan Kenyang

c. Kondisi dimana sebagian filsuf sedang makan dalam kondisi lapar dan sebagian filsuf sedang menunggu dalam kondisi lapar, dapat dilihat pada gambar 7 berikut.

p-ISSN : 2460-7363
e-ISSN : 2614-6606

Gambar 7. Simulasi Kondisi Filsuf Sebagian Dalam Keadaan Lapar & Sedang Makan

d. Kondisi dimana semua filsuf sedang menunggu dalam kondisi lapar sehingga terjadi status *deadlock*, dapat dilihat pada gambar 8 berikut.

Gambar 8. Simulasi Kondisi Semua Filsuf Dalam Keadaan Lapar & Menunggu (*Deadlock*)

3. *Banker Algorithm Problem*, fungsional yang akan ditampilkan antara lain :
   a. Kertas kerja inisialisasi awal *Banker Algorithm Problem*, dapat dilihat pada gambar 9 berikut.

Gambar 9. Kertas Kerja *Banker Algorithm Problem*

b. Tampilan Awal Simulasi *Banker Algorithm Problem*, dapat dilihat pada gambar 10 berikut.

Gambar 10. Tampilan Awal Simulasi *Banker Algorithm Problem*

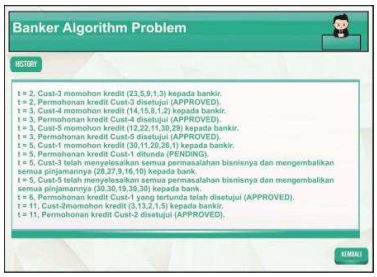
c. Simulasi *Banker Algorithm Problem (Approved)* dikarenakan kondisi *safe state*, dapat dilihat pada gambar 11 berikut.

Gambar 11. Simulasi *Banker Algorithm Problem (Approved)*

d. Simulasi *Banker Algorithm Problem (Pending)* dikarenakan *unsafe state*, dapat dilihat pada gambar 12 berikut.

Gambar 12. Simulasi *Banker Algorithm Problem (Pending)*

e. *History* Proses Simulasi *Banker Algorithm Problem*, dapat dilihat pada gambar 13 berikut.

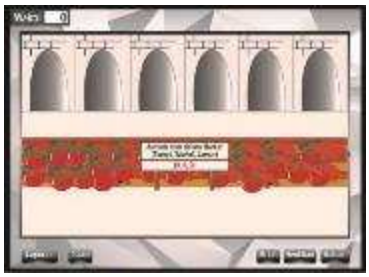


Gambar 13. *History* Proses Simulasi *Banker Algorithm Problem*

4. *Producer-Consumer Problem*, fungsional yang akan ditampilkan antara lain :
   a. Kertas Kerja inisialisasi awal *Producer-Consumer Problem*, dapat dilihat pada gambar 14 berikut.

Gambar 14. Kertas Kerja *Producer-Consumer Problem*

   b. Simulasi inisialisasi *Producer-Consumer Problem*, dapat dilihat pada gambar 15 berikut.



Gambar 15. Simulasi *Producer-Consumer Problem (Inisialisasi)*

   c. Simulasi *Producer - Consumer Problem* dengan kondisi *producer full*, dapat dilihat pada gambar 16 berikut.

Gambar 16. Simulasi *Producer-Consumer Problem (Producer Full)*

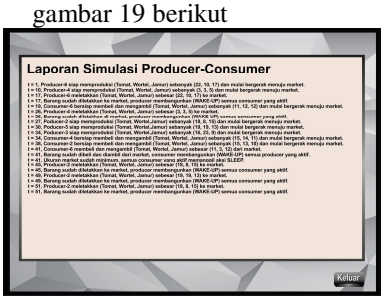
   d. Simulasi *Producer - Consumer Problem* dengan kondisi *consumer full*, dapat dilihat pada gambar 17 berikut.

Gambar 17. Simulasi *Producer-Consumer Problem (Consumer Full)*

   e. Proses Simulasi dalam bentuk Tabel pada *Producer-Consumer Problem*, dapat dilihat pada gambar 18 berikut.

Gambar 18. Proses Simulasi (Tabel) *Producer-Consumer Problem*

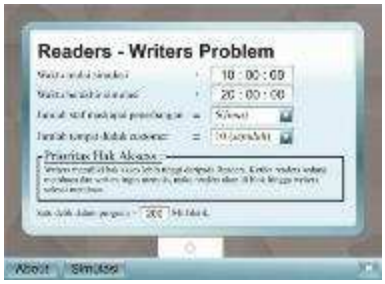
   f. *History* Proses Simulasi *Producer-Consumer Problem*, dapat dilihat pada gambar 19 berikut



Gambar 19. *History* Proses Simulasi *Producer-Consumer Problem*

5. *Readers-Writers Problem*, fungsional yang akan ditampilkan antara lain :
   a. Kertas Kerja inisialisasi awal *Readers-Writers Problem*, dapat dilihat pada gambar 20 berikut.

p-ISSN : 2460-7363
e-ISSN : 2614-6606



Gambar 20. Kertas Kerja *Readers-Writers Problem*

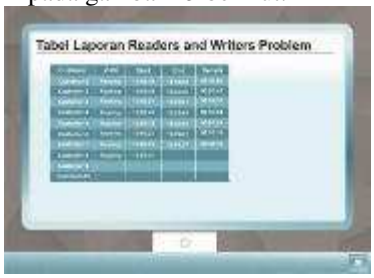
b. Proses simulasi *Readers-Writers Problem* dengan status *running readers,* dapat dilihat pada gambar 21 berikut.

Gambar 21. Proses Simulasi (*Reader Running*)

c. Proses simulasi *Readers-Writers Problem* dengan status *running writers,* dapat dilihat pada gambar 22 berikut.

Gambar 22. Proses Simulasi (*Writer Running)*

d. Proses Simulasi dalam bentuk Tabel pada *Readers-Writers Problem,* dapat dilihat pada gambar 23 berikut.



Gambar 23. Proses Simulasi (Tabel) *Readers-Writers Problem*

e. *History* Proses Simulasi *Readers-Writers Problem,* dapat dilihat pada gambar 24 berikut.

Gambar 24. *History* Proses Simulasi *Readers-Writers Problem*

VI. KESIMPULAN DAN SARAN

VI.1 Kesimpulan

Berdasarkan analisis yang dilakukan pada penelitian terhadap penerapan Simulasi Pembelajaran *Concurrency* Pada Mata Kuliah Sistem Operasi Berbasis Multimedia yang dibangun dapat ditarik kesimpulan sebagai berikut:

1. Simulasi *Dining Philosophers Problem, Banker Algorithm, Producer-Consumer Problem,* dan *Readers-Writers Problem* mampu menjelaskan secara jelas salah satu materi Sistem Operasi yaitu konkurensi, dimana di dalamnya terdapat masalah-masalah seperti *mutual exclusion, deadlock,* sinkronisasi, dan *starvation.*
2. Solusi dari setiap permasalahan pada konkurensi sudah difasilitasi oleh simulasi seperti *wait and signal, safe and unsafe state, resource request, sleep and wake up (semaphore),* dan *mutex signal*
3. Simulasi Pembelajaran *Concurrency* Pada Mata Kuliah Sistem Operasi yang sudah dikembangkan sudah memenuhi persyaratan Multimedia yaitu pemanfaatan media elektronik seperti PC dan laptop untuk mem-visualisasikan alur proses dari keempat algoritma yang dipelajari ke dalam gambar bergerak, audio, musik, *link*, dan tombol sehingga memungkinkan user berinteraksi dengan sistem

VI.2 Saran

Beberapa saran dan masukan berikut diharapkan dapat memberikan perbaikan dalam penelitian selanjutnya, yaitu:

1. Simulasi Pembelajaran Pada Mata Kuliah Sistem Operasi perlu ditambahkan materinya agar lebih lengkap sesuai dengan silabus yang sudah disusun.
2. Peningkatan simulasi dari gambar bergerak menjadi animasi yang lebih halus dilengkapi

dengan konsep / landasan teori berbasis audio visual.

3. Penambahan kertas kerja berbasis *web* untuk lebih meningkatkan daya tangkap mahasiswa memakai media tabel simulasi.
4. Penambahan *database* agar setiap aktivitas simulasi dapat tersimpan dengan baik dan dapat dipanggil sewaktu-waktu.

DAFTAR PUSTAKA

[1] Budiman, Willy. 2006. Perangkat Lunak Simulasi *Producer-Consumer Problem*. Medan: STMIK Mikrosil

[2] Canharta, Victor. 2006. *Dining Philosophers Problem Simulation*. Medan: STMIK Mikrosil

[3] Febrian, Jack. 2002. Kamus Komputer dan Istilah Teknologi Informasi. Bandung: Informatika

[4] Hofstetter. 2001. *Multimedia Literacy*. Boston: Irwin/McGraw-Hill

[5] Law, Averill M., Kelton, W. David. 1991. *Simulation Modeling & Analysis*, *Second Edition*, McGraw-Hill

[6] Maulani, G. A. F. 2016. Rancang Bangun Aplikasi Ensiklopedia Digital Tentang Tata Surya Berbasis Mobile Menggunakan J2ME. JURNAL PETIK, 2(2), 11-16.

[7] Silberschatz, Avi., Galvin, Peter., Gagne., Greg. 2007. *Operating System Concepts 8 Ed*. John Wiley & Sons, Inc.

[8] Sugiyono. 2003. Metode Penelitian Bisnis. Bandung: Alfabeta

[9] Suyanto., M. 2004. Aplikasi Desain Grafis Untuk Periklanan. Yogyakarta: Andi

[10] Tandri, Tomi. 2006. Simulasi *Readers And Writers Problem* Pada Maskapai Penerbangan Di Bandara. Medan: STMIK Mikrosil

[11] Yulianto, Erwin., Haryana, KM Syarif. 2016. Simulasi Kinematika Interaktif (Studi Kasus: Balai Diklat Metrologi). STMIK Mardira Indonesia, Bandung: Jurnal *Computech* & Bisnis

[12] Yulianto, Erwin., Ilman. Benie. 2018. Simulasi Manajemen Penempatan Barang Pada Gudang Berbasis 3D Menggunakan Metode *Block Stacking*. Universitas Langlangbuana, Bandung: Tiarsie

[13] Zhuang, Effendy. 2006. Perangkat Lunak Simulasi Algoritma *Banker*. Medan: STMIK Mikrosil