BAB II TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Penelitian ini dimaksudkan untuk menganalisis penelitian sebelumnya, yang sejalan dan mempunyai konsep yang hamper sama dengan penelitian saat ini. Beberapa penelitian terkait yang membahas sistem informasi penjadwalan adalah sebagai berikut :

Jurnal pertama dari hasil penelitian Wildan Gunardi pada tahun 2016 yang berjudul "Penjadwalan Kompetisi Sepak Bola Menggunakan Metode Erdos-Renyi Dengan K-NN dan Rule-Based Untuk Meminimalkan Biaya Operasional Klub" Penelitian ini tentang mengusulkan jadwal pembuatan skema secara otomatis berdasarkan integrasi aturan rule based, hasil dari pengujian dan perbandingan dengan jadwal dari PT. Liga Indonesia adalah enam klub (33,3 %) dari 18 peserta yang biaya pengeluarannya lebih sedikit ketika k=1 (dua klub per cluster). Ketika k=2, klub yang biaya pengeluarannya menurun adalah 14 klub (77,8 %). Kesimpulannya, semakin besar k yang ditentukan, biaya yang dikeluarkan dapat semakin ditekan (Wildan, 2013).

Jurnal kedua hasil penelitian dari Febi Andriono, Burhanuddin Dirgantara, dan Agus Virgono pada tahun 2015 yang berjudul "Perancangan Pembuatan Dan Analisa Sistem Informasi Penyusunan Jadwal Kuliah Berbasis Web Dengan Kecerdasan Buatan" penelitian ini membahas tentang permasalahan Penyusunan jadwal kuliah merupakan masalah yang cukup kompleks yang dihadapi oleh suatu perguruan tinggi. Jadwal kuliah yang tersusun harus memperhatikan kepentingan dosen dan mahasiswa serta peraturan yang ada. Jika penyusunan jadwal kuliah dilakukan oleh manusia, akan membutuhkan waktu, tenaga, dan pikiran yang tidak sedikit. Selain itu juga diperlukan pengalaman yang cukup. Dengan mengkombinasikan antara algoritma RBS dengan algoritma genetika (hybrid), maka performansi sistem untuk mengolah jadwal kuliah jauh lebih baik dibandingkan menggunakan algoritma genetika murni. Terbukti dengan menggunakan algoritma hybrid nilai fitness terbaik yaitu 2088 diperoleh pada iterasi ke-500. Sedangkan pada iterasi tersebut, algoritma genetika

masih menghasilkan nilai fitness terbaik sebesar 36152 dan pada iterasi ke-1000 masih menghasilkan nilai fitness terbaik sebesar 10113 (Andriono, Dirgantara, & Virgono, 2015).

Jurnal ketiga dari hasil penelitian Khafidh Fidiansyah Nugroho pada tahun 2018 yang berjudul "Implementasi Rule Based Untuk Menentukan Efektifitas Antibiotik Terhadap Bakteri Pada Game 3D Adventure Of Antibod" penelitian ini membahas tentang Permasalahan kebebasan jual beli antibiotic dipasaran yang terkadang tak dimanfaatkan secara bijaksana, dimana bebas membeli bukan berarti dapat menggunakan secara bebas. Penggunaan metode rule based sangat cocok pada pembangunan game ini, Media game edukasi dipilih karena dirasa mudah dipahami dan menyenangkan. Sehingga membuat penggunanya tak merasa sedang melakukan pembelajaran ketika bermain. Selain itu game juga dapat menjangkau semua usia. Pembelajaran melaui game juga lebih mudah dipahami ketimbang membaca buku ataupun mendengarakan pembelajaran. Penggunaan metode Rule Based System dirasa cocok pada pembangunan game ini. Penggunaan Ruled Based System dipilih karena memiliki akurasi yang sangat bagus. Dengan aturan rule based foward chaining, aturan akan dieliminasi hingga premis yang masuk kriteria dieksekusi. Sehinga miss informasi dalam game dapat diminimalisir hingga 0% (Nugroho, 2018).

2.2 Penjadwalan

Penjadwalan (*Scheduling*) atau membuat Jadwal adalah salah satu kegiatan yang penting dalam proses produksi ataupun pekerjaan suatu proyek. Penjadwalan digunakan sebagai dasar untuk mengalokasikan sumber daya pabrik seperti mesin dan peralatan produksi, merencanakan sumber daya manusia yang akan digunakan, pembelian material dan merencanakan proses produksi. Penjadwalan yang baik akan memberikan dampak yang positif terhadap kelancaran produksi serta meminimalkan waktu dan biaya produksi (Herjanto, 2001).

2.3 Rule Base Generator

Rule based generator (RBG) merupakan suatu sistem pakar yang menggunakan aturan-aturan untuk menyajikan pengetahuannya. Menurut Lusiani dan Cahyono, sistem berbasis aturan adalah suatu perangkat lunak yang menyajikan keahlian pakar dalam bentuk aturan-aturan pada suatu domain tertentu untuk menyelesaikan suatu permasalahan. RBG adalah model sederhana yang dapat diadaptasi ke banyak masalah. Namun, jika aturan terlalu banyak, pemeliharaan sistem akan rumit dan terdapat banyak kesalahan dalam kerjanya.

Untuk membuat RBG, ada beberapa hal penting yang harus dimiliki:

1. Sekumpulan fakta yang mewakili *working memory*. Ini dapat berupa suatu keadaan yang relevan dengan keadaan awal sistem bekerja.
2. Sekumpulan aturan. Aturan ini mencakup setiap tindakan yang harus diambil dalam ruang lingkup permasalahan yang dibutuhkan.
3. Kondisi yang menentukan bahwa solusi telah ditemukan atau tidak (*non-exist*). Hal ini berguna untuk menghindari *looping* yang tidak akan pernah berakhir.

Teori RBG ini menggunakan teknik yang sederhana, dimulai dengan dasar aturan yang berisi semua pengetahuan dari permasalahan yang dihadapi yang kemudian dikodekan ke dalam aturan *if-then* yang mengandung data, pernyataan dan informasi awal. Sistem akan memeriksa semua aturan kondisi *if* yang menentukan subset, set konflik yang ada. Jika ditemukan, maka sistem akan melakukan kondisi *then*. Perulangan ini akan terus berlanjut hingga salah satu atau dua kondisi bertemu, jika aturan tidak diketemukan maka sistem tersebut harus keluar dari perulangan. Untuk dibangun, *Rule Base Generator* setidaknya memiliki tiga elemen. (Lusiani & Cahyono, 2006).

1. Kumpulan data dan fakta. Kumpulan fakta disini sebagai acuan dasar dan acuan pengetahuan yang kemudian akan diproses menjadi aturan

2. Kumpulan *rule*/aturan. Fakta – fakta yang sudah terkumpul akan dirangkai menjadi suatu aturan yang bias dimengerti mesin.

3. Kriteria untuk mengakhiri. Kriteria disini adalah sebuah kondisi yang dijadikan acuan pada sistem. Sehingga nantinya dapat ditentukan apakah sistem akan berhenti atau justru terus melakukan looping. Berikut merukapan contoh algoritma transformasi data menjadi rule pada Gambar 2.1.

DATA	CONDITIONS	RULES
Season Winter temperature wind blushing road weather	<0, >0 strongly, gently slippery, not slippery cold, warm, hot	***Premises*** IF temperatur < 0 AND IF wind blushing is strongly OR IF the road is slippery ***Conclusion*** THEN the weather is cold

Gambar 2. 1 Contoh algoritma transformasi data menjadi *rule*

Keterangan :

a. Pada kolom data ditemukan sejumlah fakta dan data mengenai temperature, kecepatan angina, kondisi jalan dan cuaca
b. Data-data yang tersedia kemudian diberikan sebuah kondisi yang mampu merujuk pada sebuah konklusi.
c. Setelah selesai data yang terkondisi kemudian dimasukkan dalam bentuk *if-then*, dimana *If* berisikan (data + kondisi) dan *Then* berisikan konklusi.

2.4 *Unified Modeling Language*

Unified Modeling Language (UML) adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membanngun perangkat lunak. UML merupakan *metodologi* dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem (Ade , 2016).

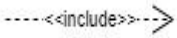
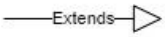
Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut:

a. *Use Case Diagram*

Use Case diagram merupakan pemodelan untuk kelakuakn (*behavior*) sistem informasi yang akan dibuat. *Use Case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut Ade Hendini (2016). Simbol-simbol yang digunakan dalam *Use Case diagram* terdapat pada Tabel 2.1.

Tabel 2. 1 Simbol *Use Case Diagram*

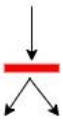
Simbol	Keterangan
	*Use Case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktir, yang dinyatakan dengan menggunakan kata kerja.
	*Actor* atau Aktor menggambarkan *abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktor, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem dapat muncul dalam beberapa peran. Aktor berinteraksi dengan *Use Case*, tetapi tidak memiliki kontrol terhadap *Use Case*.
	Asosiasi antara aktor dan *Use Case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengindikasikan data.

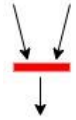
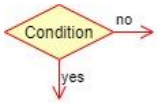
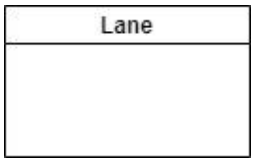
	Asosiasi antara aktor dan *Use Case* yang menggunakan panah terbuka untuk mengindikasikan bila aktor berinteraksi secara pasif dengan sistem.
	*Include* menggambarkan suatu simbol di dalam simbol lain (*required*) atau pemanggilan *Use Case* oleh *Use Case* lain, contohnya adalah pemanggilan sebuah fungsi program.
	*Extend* menggambarkan perluasan dari simbol lain jika kondisi atau syarat terpenuhi.

b. *Activity Diagram*

Activity diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis Ade Hendini (2016). Simbol-simbol yang digunakan dalam *activity diagram* terdapat pada Tabel 2.2.

Tabel 2. *2 Activity Diagram*

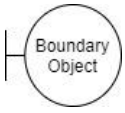
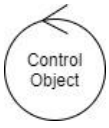
Simbol	Keterangan
	*Start point* menggambarkan awal dari sebuah aktivitas.
	*End point* menggambarkan akhir dari sebuah aktifitas.
	*Activities* menggambarkan suatu proses atau kegiatan bisnis.
	*Fork* atau percabangan digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau bersamaan.

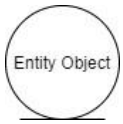
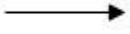
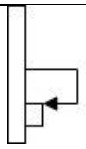
	*Join* atau penggabungan digunakan untuk menunjukkan adanya dekomposisi.
	*Decision points* menggambarkan pilihan untuk pengambilan keputusan yaitu *true* atau *false*.
	*Swimlane* menggambarkan pembagian *activity diagram* untuk menunjukkan siapa melakukan apa.

c. *Sequence Diagram*

Sequence Diagram menggambarkan kelakuan objek pada *Use Case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek (Ade , 2016). Simbol-simbol yang digunakan dalam *Sequence Diagram* terdapat pada Tabel 2.3.

Tabel 2. 3 Simbol *Sequence Diagram*

Simbol	Keterangan
	*Boundary class* merupakan kumpulan kelas yang menjadi interfaces atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan *form entry* dan *form* cetak.
	*Control class* merupakan suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas.

	*Entity class* merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.
	Message digunakan mengirim pesan antar *class*.
	*Recursive* menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri
	Activation, mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivasi sebuah operasi
	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*

d. *Class Diagram*

Class diagram merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem. *Class diagram* juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan (Ade , 2016).

Class diagram secara khas meliputi: kelas (*class*), relasi *assosiations*, *generalitation* dan *aggregation*, attribut (*attributes*), operasi (*operation* atau *method*) dan *visibility*, tingkat akses objek eksternal kepada suatu operasi atau attribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau *cardinality* (Ade , 2016). Jenis dari *multiplicity* terdapat pada Tabel 2.4.

Tabel 2. *4 Multiplicity Class Diagram*

Multiplicity	Keterangan
1	Satu dan hanya satu
0...*	Boleh tidak ada atau satu atau lebih
1…*	Satu atau lebih
0…1	Boleh tidak ada dan maksimal satu
n…n	Batasan antara. Contoh 2…4, memiliki arti minimal 2 maksimal 4.