BAB 3
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Masalah

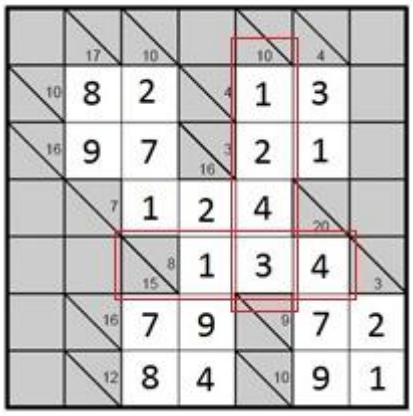
Kakuro adalah permainan *puzzle* yang bersifat logika dan biasa disebut sebagai transliterasi matematis dari *puzzle* silang. Permainan ini adalah permainan penjumlahan angka dengan aturan mainnya yaitu dengan mengisi setiap kotak yang tersedia (kotak jawaban) dengan angka bulat dari 1 hingga 9 dengan beberapa syarat dalam pengisiannya. Soal-soal dalam permainan *puzzle* kakuro bersifat unik, karena soal-soal ini akan saling terkait baik dengan soal lainnya maupun jawaban dalam menyelesaikan permainan *puzzle* kakuro [5]. Untuk menyelesaikan permainan *puzzle* kakuro ini, jumlah total pada nilai jawaban harus bernilai sama dengan nilai soal dan tidak boleh terdapat dua angka yang sama pada satu lajur (kolom maupun baris).

	17	10		10	4	
10	8	2	4	1	3	
16	9	7	3 16	2	1	
	7	1	2	4	20	
		8 15	1	3	4	3
	16	7	9	9	7	2
	12	8	4	10	9	1

Gambar 3.1 Contoh penyelesaian *puzzle* kakuro

Oleh karena itu dibutuhkan teknik untuk membangkitkan soal yang bersifat unik dengan sistem *random* yang akan menghasilkan soal yang akan saling terkait dari soal satu dengan soal lainnya dan mencari solusi dalam menyelesaikan jawaban pada permainan *puzzle* kakuro dengan beberapa syarat didalamnya. Dengan menggunakan prinsip dasarnya, algoritma *backtracking* mencari segala kemungkinan solusi dengan menerapkan ruang keadaan dalam bentuk pohon

solusi hingga menghasilkan solusi yang layak. Kelayakan solusi pada permainan *puzzle* kakuro didefinisikan dengan suatu nilai akan dinyatakan layak apabila tidak terdapat nilai angka yang sama pada satu serta nilai total pada jawaban harus bernilai sama dengan nilai soal. Fungsi solusi didefiniskan jika semua kotak kosong (kotak jawaban) telah terisi dengan nilai yang sesuai (sesuai dengan fungsi kelayakan). Berikut gambaran kelayakan solusi pada permainan *puzzle* kakuro :



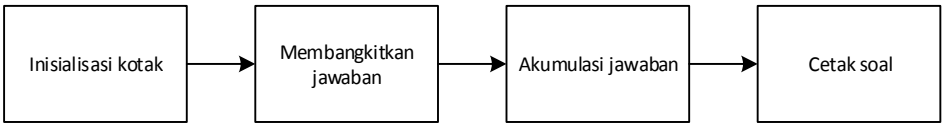
Gambar 3.2 Solusi layak pada permainan *puzzle* kakuro

3.2 Analisis *Game*

Analisis *game* yang akan diteliti pada *game puzzle* kakuro adalah dengan membangkitkan soal permainan serta menyelesaikan jawaban. Pada dasarnya permainan *puzzle* kakuro ini disetiap soal maupun jawaban akan saling terkait satu sama lainnya dan mempunyai nilai pasti disetiap jawabannya. Oleh karena itu, algoritrma *backtracking* akan diterapkan pada kasus *puzzle* kakuro ini. Berikut merupakan alur membangkitkan soal permainan serta penyelesaian jawaban :

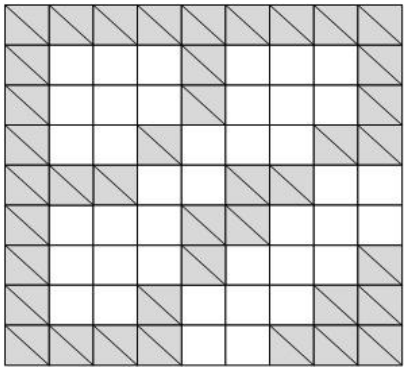
1. Proses pembangkitan soal.

   Pada proses pembangkitan soal, nilai soal harus saling terkait dengan nilai soal lainnya. Pada kasus ini, algoritma *backtracking* akan diterapkan dalam membangkitkan soal. Algoritma *backtracking* bekerja untuk mencari nilai yang akan menjadi nilai jawaban dalam pembangkitan soal dengan menerapkan aturan permainan pada *puzzle* kakuro yaitu tidak boleh terdapat angka yang sama dalam satu lajur.



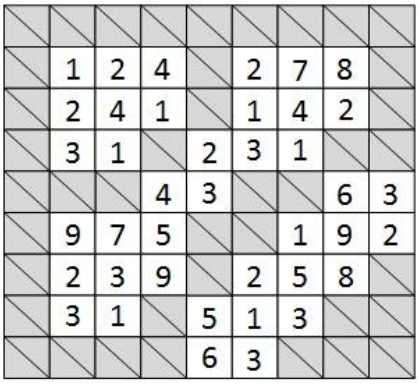
Gambar 3.3 Proses pembangkitan soal

Berikut penjelasan alur proses pada gambar 3.3 dengan kasus permainan *puzzle* kakuro :



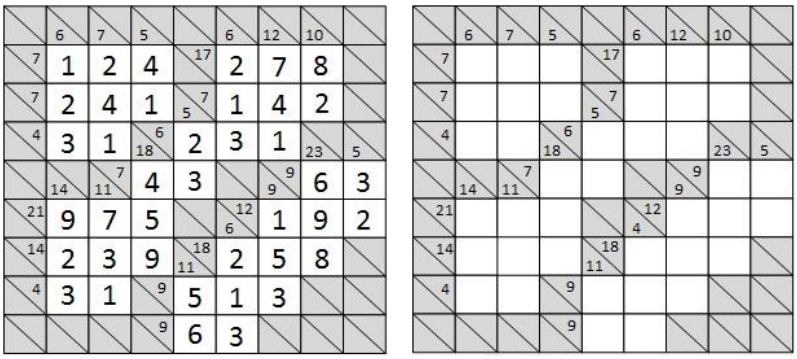
Gambar 3.4 Inisialisasi kotak permainan

Gambar 3.4 menjelaskan tentang keadaan awal permainan sebelum menentukan nilai acuan pada lajur. Dimana kotak permainan disusun berdasarkan kotak blok yang berwarna hitam dan kotak kosong yang berwarna putih.



Gambar 3.5 Membangkitkan jawaban

Gambar 3.5 menjelaskan tentang pembangkit nilai jawaban. Pengisian ini bertujuan untuk membangkitkan nilai soal pada kotak blok. Selain bertujuan mencari nilai soal proses ini juga bertujuan untuk menentukan jawabannya.

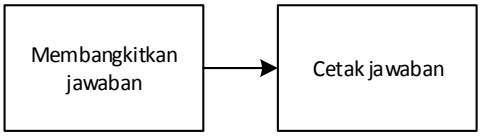


Gambar 3.6 Akumulasi jawaban dan cetak soal

Gambar 3.6 menjelaskan tentang akumulasi nilai jawaban dan cetak nilai soal untuk permainan ini. Untuk mencetak soal, kotak yang berisikan nilai jawaban dalam satu lajur akan diakumulasikan dari total nilai jawaban.

2. Proses penyelesaian jawaban pada kotak kosong.

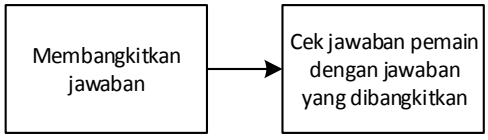
   Pada proses ini, penyelesaian jawaban dengan algoritma *backtracking* dengan sistem terurut dalam pencarian nilai. Proses pencarian nilai dilakukan dari angka 1 hingga 9, dan dengan melakukan tahapan pengecekan nilai yaitu tidak boleh terdapat angka yang sama pada satu lajur serta total nilai jawaban pada satu lajur yang akan dibandingkan dengan nilai soal.



Gambar 3.7 Proses penyelesaian jawaban pada kotak kosong

3. Proses penyelesaian jawaban pada kotak terisi nilai.

   Pada proses ini, penyelesaian jawaban dengan algoritma *backtracking* hampir sama dengan proses penyelesaian jawaban pada kotak tidak terisi. Hanya saja pada proses pengecekan nilai, nilai jawaban yang telah diisi oleh pemain akan dicek ulang dengan nilai jawaban pada algoritma *backtracking*.



Gambar 3.8 Proses penyelesaian jawaban pada kotak terisi nilai

3.3 Analisis Masukan

Analisis masukan yang dilakukan dalam penelitian ini adalah menganalisis inputan *puzzle* dalam algoritma *backtracking* pada *game* kakuro. Masukan tersebut berupa nilai (angka), akan tetapi masukan diberi batasan nilai yaitu dari angka 1 hingga 9, berikut merupakan tabel analisis masukan pada *game puzzle* kakuro.

Tabel 3.1 Nilai masukan 2 kotak

Nilai acuan lajur	Nilai kombinasi
3	1-2
4	1-3
5	1-4; 2-3
6	1-5; 2-4
7	1-6; 2-5; 3-4
8	1-7; 2-6; 3-5
9	1-8; 2-7; 3-6; 4-5
10	1-9; 2-8; 3-7; 4-6
11	2-9; 3-8; 4-7; 5-6
12	3-9; 4-8; 5-7
13	4-9; 5-8; 6-7
14	5-9; 6-8
15	6-9; 7-8
16	7-9
17	8-9

Tabel 3.2 Nilai masukan 3 kotak

Nilai acuan lajur	Nilai kombinasi
6	1-2-3
7	1-2-4
8	1-2-5; 1-3-4
9	1-2-6; 1-3-5; 2-3-4

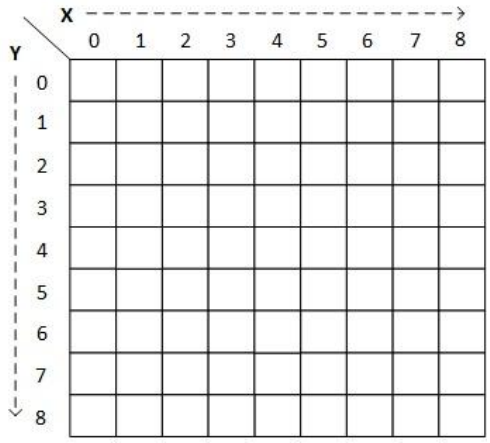
Nilai acuan lajur	Nilai kombinasi
10	1-2-7; 1-3-6; 1-4-5; 2-3-5
11	1-2-8; 1-3-7; 1-4-6; 2-3-6; 2-4-5
12	1-2-9; 1-3-8; 1-4-7; 1-5-6; 2-3-7; 2-4-6; 3-4-5
13	1-3-9; 1-4-8; 1-5-7; 2-3-8; 2-4-7; 2-5-6; 3-4-6
14	1-4-9; 1-5-8; 1-6-7; 2-3-9; 2-4-8; 2-5-7; 3-4-7; 3-5-6
15	1-5-9; 1-6-8; 2-4-9; 2-5-8; 2-6-7; 3-4-8; 3-5-7; 4-5-6
16	1-6-9; 1-7-8; 2-5-9; 2-6-8; 3-4-9; 3-5-8; 3-6-7; 4-5-7
17	1-7-9; 2-6-9; 2-7-8; 3-5-9; 3-6-8; 4-5-8; 4-6-7
18	1-8-9; 2-7-9; 3-6-9; 3-7-8; 4-5-9; 4-6-8; 5-6-7
19	2-8-9; 3-7-9; 4-6-9; 4-7-8; 5-6-8
20	3-8-9; 4-7-9; 5-6-9; 5-7-8
21	4-8-9; 5-7-9; 6-7-8
22	5-8-9; 6-7-9
23	6-8-9
24	7-8-9

3.4 Analisis Algoritma

Analisis algoritma yang akan diteliti pada *game puzzle* kakuro ini adalah untuk menganalisis cara kerja algoritma *backtracking* terhadap kasus permainan *puzzle* kakuro, bagaimana cara kerja algoritma *backtracking* dalam pembangkitan soal serta penyelesaian jawaban pada *game puzzle* kakuro.

Sebelum meneliti cara kerja algoritma *backtracking* pada *game puzzle* kakuro ini, akan dibahas terlebih dahulu bagaimana cara menginisialisasikan kotak terlebih dahulu yang akan menghasilkan kotak kosong (kotak jawaban) dan kotak blok (kotak nilai soal / pembatas).

Dalam proses inisialisasi kotak pada *puzzle* kakuro, di setiap kotak *puzzle* kakuro akan diberikan nilai indeks yang akan digambarkan dengan titik koordinat sumbu (X, Y). Pada sumbu X menyatakan posisi kotak secara horizontal, dan sumbu Y menyatakan posisi kotak secara vertikal. Titik koordinat sumbu X, dan Y akan di gambarkan sebagai berikut :



Gambar 3.9 Titik koordinat sumbu (X, Y)

Pada gambar 3.9 menjelaskan keadaan awal kotak sebelum terjadinya inisialisasi kotak permainan yang akan menentukan nilai acuan lajur dan kotak permainan. Dimana kotak permainan disusun berdasarkan kotak blok yang berwarna hitam dan kotak kosong yang berwarna putih seperti yang terjadi pada proses gambar 3.4. Dalam menentukan kotak blok dan kosong dalam permainan *puzzle* kakuro ini, maka alur proses yang akan terjadi adalah sebagai berikut :

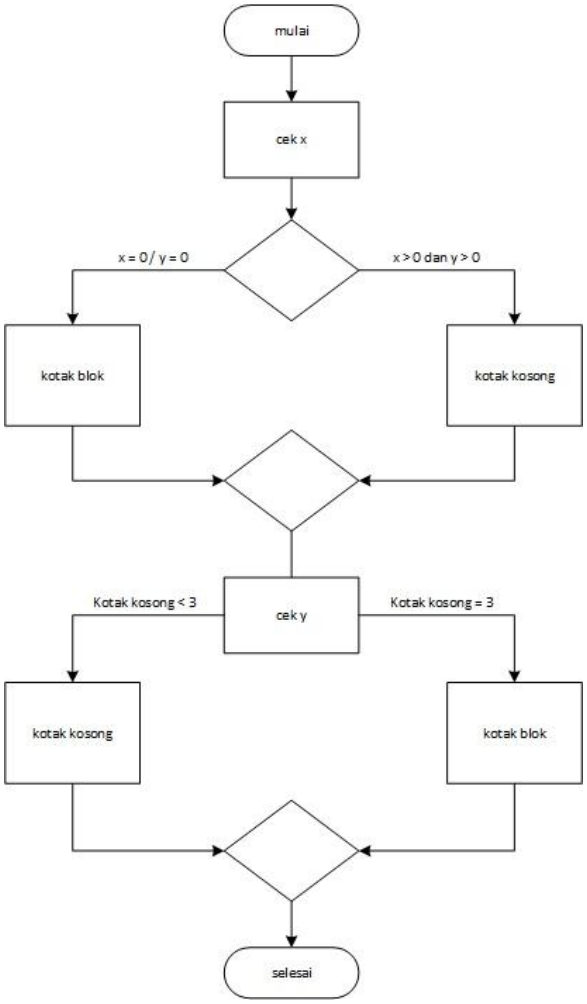
1. Mengecek lajur nilai X.

   Apabila nilai indeks pada kotak yang dipilih terdapat nilai 0 pada sumbu X atau Y, maka kotak tersebut akan menjadi kotak blok. Sedangkan apabila nilai indeks pada kotak yang dipilih terdapat nilai > 0 pada sumbu X atau Y, maka kotak tersebut akan menjadi kotak kosong.

2. Mengecek lajur nilai Y.

   Nilai indeks pada kotak yang dipilih akan menjadi kotak blok, apabila lajur pada sumbu Y diatasnya terdapat 3 kotak kosong. Sedangkan nilai indeks pada kotak yang dipilih akan menjadi kotak kosong, apabila lajur pada sumbu Y diatasanya kurang dari 3 kotak kosong. Pada pengecekan lajur sumbu Y terdapat batasan, karena pada permainan yang dibuat hanya dibatasi 2 hingga 3 kotak kosong sebagai nilai masukan.

Dari pengecekan tersebut, didapatlah kotak blok dan kosong untuk menentukan lajur permainan seperti pada gambar gambar 3.4. Proses pengecekan nilai pada titik koordinat (X, Y) diatas akan dijelaskan pada alur *flowchart* berikut :



Gambar 3.10 *Flowchart* pengecekan nilai pada titik koordinat (X, Y)

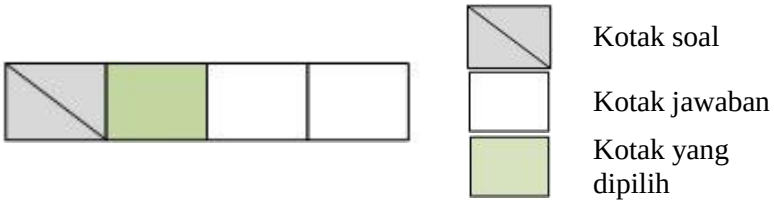
3.4.2 Analisis Algoritma *Backtracking* Pembangkit Nilai Soal

Analisis algoritma *backtracking* pada pembangkit nilai soal merupakan penjelasan alur atau cara kerja algoritma *backtracking* saat terjadi pembuatan soal di dalam *game puzzle* kakuro yang sudah terinisialisasi kotak terlebih dahulu. Algoritma *backtracking* akan bekerja saat pencarian nilai dan akan melakukan *backtrack* apabila terdapat nilai angka yang sama dalam satu lajur. Penjelasan tersebut akan digambarkan pada kotak sebagai berikut.



Gambar 3.11 Keadaan awal satu lajur

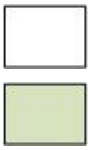
Gambar 3.11 menjelaskan tentang keadaan awal satu lajur sebelum menentukan nilai soal. Dimana kotak dari satu lajur disusun berdasarkan kotak jawaban dan kotak soal. Setelah tampil satu lajur tersebut, alur kerja algoritma *backtracking* akan dimulai dengan menujuk kotak jawaban pertama menggunakan *pointer* yang akan dicari nilai jawaban kedalam *array.*



Gambar 3.12 Memilih kotak jawaban

Gambar 3.12 menjelaskan ketika *pointer* memilih kotak jawaban. Sebelum menentukan nilai jawaban, kotak jawaban di cek terlebih dahulu oleh sistem apakah kotak jawaban dalam kondisi kosong atau terisi nilai dalam satu lajur. Apabila kotak jawaban dalam kondisi kosong, maka sistem akan mengisi kotak jawaban dengan cara *random.* Dan apabila kotak jawaban dalam kondisi terisi nilai maka *pointer* akan berpindah ke kotak jawaban yang lain.





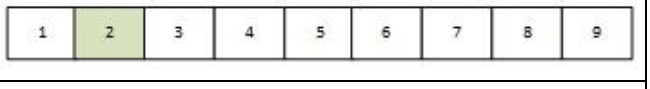
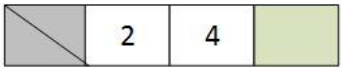
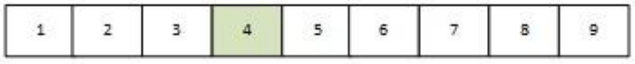
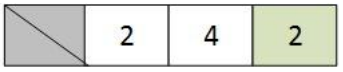
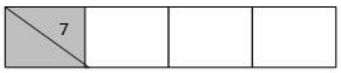
Kotak masukan nilai kakuro

Kotak yang dipilih oleh *pointer*

Gambar 3.13 Pencarian nilai masukan di dalam *array*

Gambar 3.13 menjelaskan pencarian nilai masukan untuk kotak jawaban di dalam *array*. Pencarian dilakukan berdasarkan nilai yang telah ditentukan yaitu nilai 1 hingga 9. Penentuan pencarian nilai dilakukan dengan cara *random*. Setelah nilai dipilih maka nilai pada *array* akan di masukan kedalam kotak jawaban. Berikut terjadinya proses pembentukan nilai soal dengan menentukan nilai jawaban terlebih dahulu menggunakan algoritma *backtracking* pada tabel 3.3.

Tabel 3.3 Proses pembangkitan nilai soal

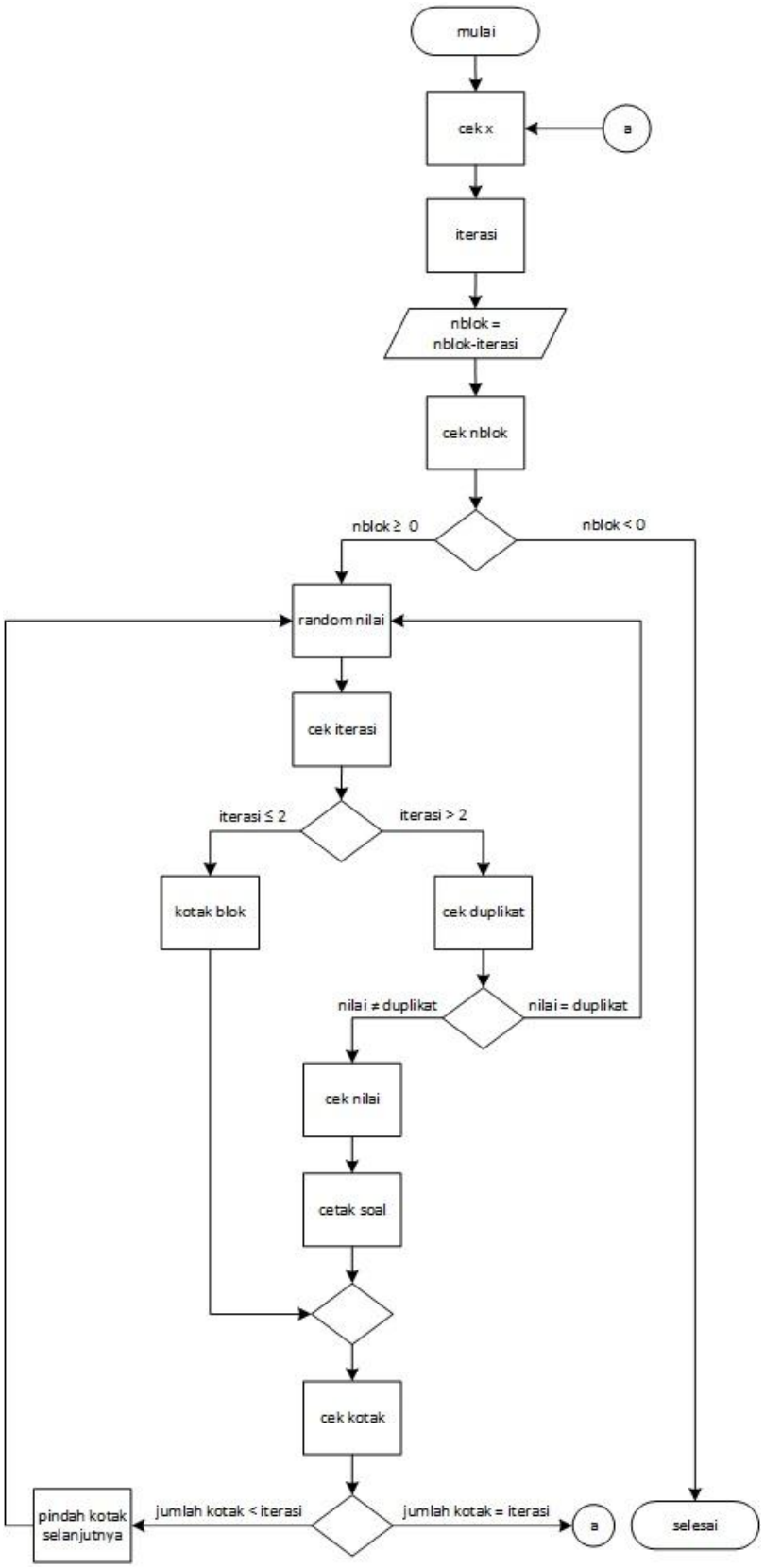
Proses	Nilai kotak jawaban	Nilai masukan *array*
1.		
2.		
3.		
4.		
5.		-

Dari proses tabel 3.3 terdapat proses-proses yang menghasilkan nilai jawaban hingga terjadinya pembangkitan nilai soal. Dari proses tersebut akan dijelaskan kembali dengan alur kerja bagaimana terjadinya proses pembangkitan nilai soal pada tabel 3.4

Tabel 3.4 Alur kerja pembangkitan nilai soal

Kotak jawaban pertama pada satu lajur	
Cek lajur	1. Apabila lajur dalam keadaan terisi nilai semua, maka proses cek lajur dilanjutkan ke lajur selanjutnya. 2. Apabila lajur dalam keadaan kosong, maka lakukan pencarian nilai jawaban.
Random nilai	Pencarian nilai jawaban yang dilakukan oleh sistem secara *random* didalam *array,* dengan ketentuan nilai dari 1 hingga 9.
Cetak nilai	Nilai yang terpilih oleh sistem pada proses pencarian nilai akan dimasukan kedalam kotak jawaban.
Kotak jawaban kedua hingga terakhir pada satu lajur	
Cek lajur, Random nilai	
Cek duplikat	1. Apabila terdapat nilai yang sama pada lajur dalam proses pencarian nilai, maka dilakukan pencarian nilai ulang oleh sistem dengan tidak memilih kembali nilai yang sudah terpilih sebelumnya. 2. Apabila nilai yang terpilih tidak bernilai sama pada lajur, maka proses pencarian nilai dilanjutkan ke kotak kosong sebelahnya.
Cetak nilai	
Cetak soal	Lajur yang telah terisikan nilai jawaban akan diakumulasikan yang akan menghasilkan nilai soal. Nilai jawaban yang sudah melalui tahap akumulasi hingga menghasilkan nilai soal, akan dihapus baik dalam kotak jawaban maupun pada sistem.

Dari alur kerja pembangkitan nilai soal tersebut, didapatlah nilai soal dari hasil pencarian nilai jawaban yang sudah diakumulasikan. Alur kerja dalam pembangkitan nilai soal diatas akan digambarkan pada alur *flowchart* berikut :



Gambar 3.14 *Flowchart* pembangkitan nilai soal

3.4.3 Analisis Algoritma *Backtracking* Penyelesaian Jawaban

Analisis algoritma *backtracking* dalam menyelesaikan jawaban terjadi kedalam dua kondisi, yaitu kondisi disaat kotak jawaban kosong atau kondisi disaat kotak jawaban terisi nilai.

3.4.3.1 Penyelesaian Jawaban Pada Kotak Kosong

Algoritma *backtracking* pada kondisi kotak jawaban kosong atau tidak terisikan nilai, algoritma *backtracking* bekerja dengan mengecek nilai jawaban dengan mengecek apakah terdapat nilai yang sama pada lajur serta mengecek akumulasi total nilai jawaban dengan dengan nilai soal. Untuk menjelaskan bagaimana algoritma *backtracking* bekerja, maka akan dijelaskan alur proses aktivitas algoritma *backtracking* yang bekerja pada tabel 3.5.

Tabel 3.5 Proses terjadinya nilai jawaban

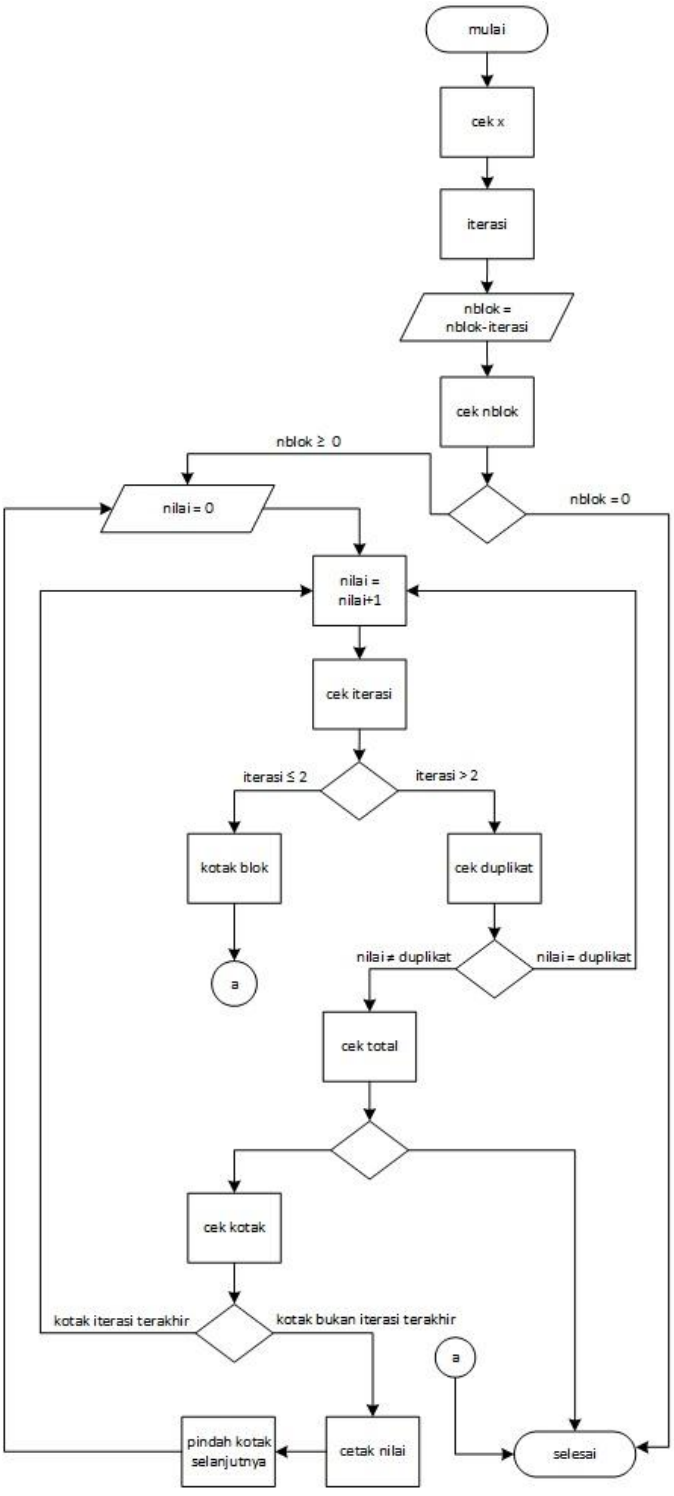
Proses	Nilai kotak jawaban	Nilai masukan *array*
1.	7	1 2 3 4 5 6 7 8 9
2.	7 1	1 2 3 4 5 6 7 8 9
3.	7 1 2	X 2 3 4 5 6 7 8 9
4.	7 1 2 3	X X 3 4 5 6 7 8 9
5.	7 1 2 4	X X X 4 5 6 7 8 9

Dari proses tabel 3.5 terdapat proses-proses yang menghasilkan nilai jawaban dari keadaan awal lajur yang sudah terisikan nilai soal. Dari proses tersebut akan dijelaskan kembali dengan alur kerja bagaimana terjadinya proses terjadinya nilai jawaban yang akan bernilai sama dengan nilai soal pada tabel tabel 3.6.

Tabel 3.6 Alur kerja terjadinya nilai jawaban

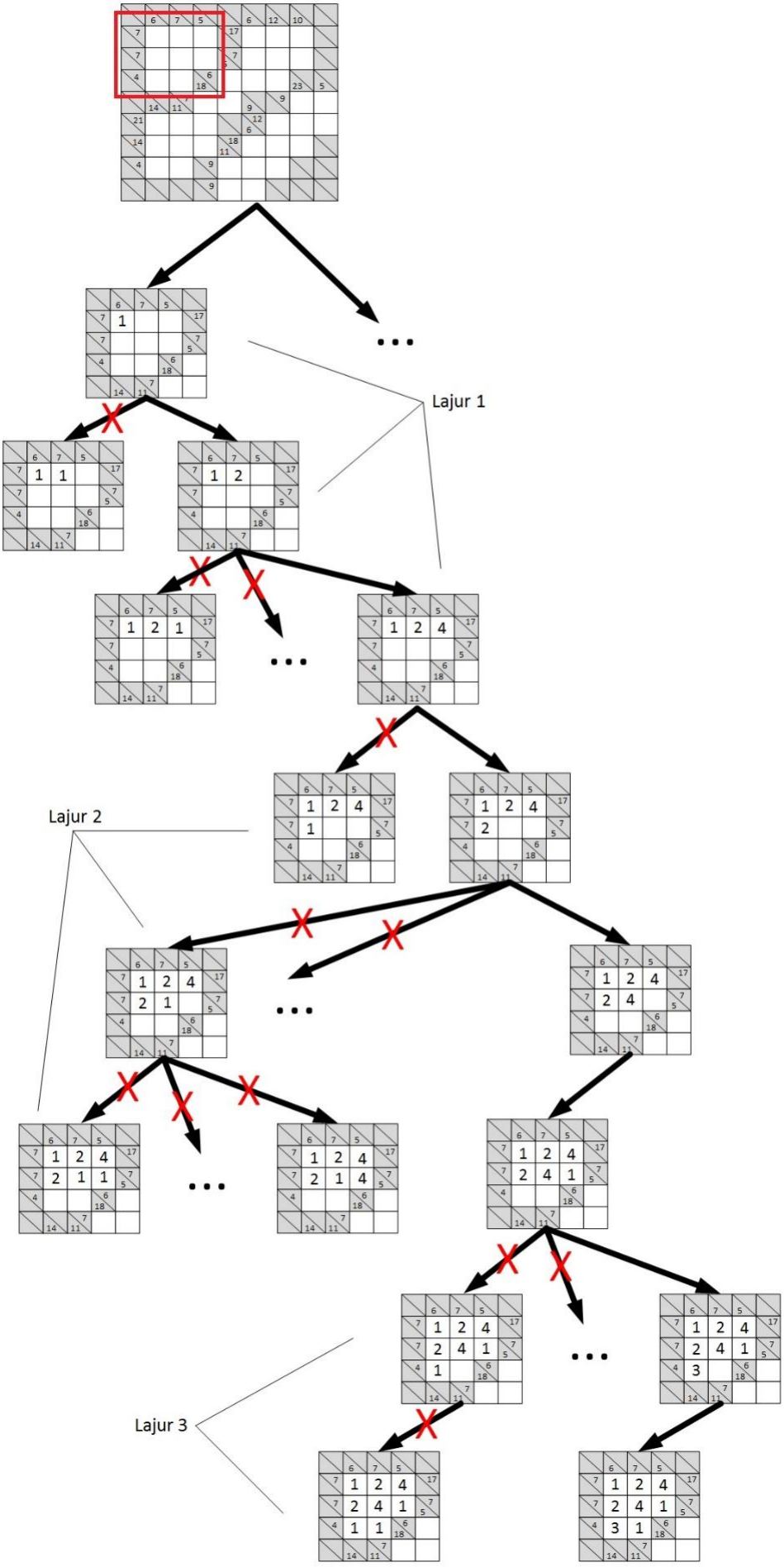
Kotak jawaban pertama pada satu lajur	
Cek lajur	1. Apabila lajur dalam keadaan terisi nilai semua, maka proses cek lajur dilanjutkan ke lajur selanjutnya. 2. Apabila lajur dalam keadaan kosong, maka lakukan pencarian nilai jawaban.
Pencarian nilai	Pencarian nilai jawaban yang dilakukan oleh sistem secara berurutan didalam *array*, dengan ketentuan nilai dari 1 hingga 9.
Cek total	1. Apabila nilai jawaban yang terpilih oleh sistem bernilai kurang dari nilai soal dan tidak berada pada kotak terakhir pada lajur, maka proses pencarian nilai dilanjutkan ke kotak kosong sebelahnya. 2. Apabila nilai jawaban yang terpilih oleh sistem bernilai kurang dari nilai soal dan berada pada kotak terakhir pada lajur, maka proses pencarian nilai dilakukan ulang dengan melanjutkan nilai sebelumnya yang terpilih hingga menghasilakn nilai yang sama dengan nilai soal apabila diakumulasikan.
Cetak nilai	Nilai yang terpilih oleh sistem pada proses pencarian nilai akan dimasukan kedalam kotak jawaban.
Kotak jawaban kedua hingga terakhir pada satu lajur	
Cek lajur, Pencarian nilai	
Cek duplikat	1. Apabila terdapat nilai yang sama pada lajur pada proses pencarian nilai, maka dilakukan pencarian nilai ulang oleh sistem dengan tidak memilih kembali nilai yang sudah terpilih sebelumnya. 2. Apabila nilai yang terpilih tidak bernilai sama pada lajur, maka proses pencarian nilai dilanjutkan ke kotak kosong sebelahnya.
Cek total, Cetak nilai	

Dari alur kerja terjadinya nilai jawaban tersebut, didapatlah nilai jawaban dari keadaan awal lajur yang sudah terisikan nilai soal. Alur kerja dalam terjadinya nilai jawaban diatas akan digambarkan pada alur *flowchart* berikut :



Gambar 3.15 Proses penyelesaian jawaban pada kotak kosong

Gambar 3.15 adalah proses penyelesaian jawaban pada kotak kosong yang digambarkan kedalam *flowchart*, dimana pada prosesnya terdapat beberapa pengecekan baik pengecekan nilaerri maupun lajur. Pengecekan nilai maupun lajur akan menghasilkan solusi yang layak, apabila memenuhi fungsi kelayakan dalam permainan *puzzle* kakuro. Solusi yang layak dalam permainan *puzzle* kakuro adalah apabila nilai jawaban dalam satu lajur bernilai sama dengan nilai soal serta nilai jawaban saling terkait dengan nilai jawaban lainnya pada lajur yang berbeda. Berikut merupakan contoh kasus dari solusi layak dalam permainan *puzzle* kakuro pada 3 lajur yang digambarkan kedalam pohon solusi pada gambar 3.16.



Gambar 3.16 Solusi layak dalam pohon solusi pada 3 lajur

Pada gambar 3.16 terjadi alur proses dalam menemukan solusi yang layak yang digambarkan kedalam pohon solusi dengan menghasilkan kedalaman 8 untuk mencapai solusi pada tiga lajur. Kedalaman pohon solusi pada permainan *puzzle* kakuro ini didapat, apabila kotak jawaban pada satu lajur tidak saling terkait lagi dengan kotak jawaban pada lajur lainnya. Dalam prosesnya, terjadi pemangkasan *(pruning)* terhadap simpul-simpul yang tidak mengarah ke solusi. Simpul yang menjadi simpul anak dan turunan tidak akan diproses kembali, karena telah terjadi pemangkasan.

3.4.3.2 Penyelesaian Jawaban Pada Kotak Terisi Nilai

Algoritma *backtracking* pada kondisi kotak jawaban terisi nilai, alur proses kerja algoritmanya sama dengan alur proses kerja algoritma *backtracking* penyelesaian jawaban pada kotak kosong yang terjadi pada tabel 3.5. Alur kerja sistem pada penyelesaian jawaban pada kotak terisikan nilai seperti pada gambar 3.8. Pada alur tersebut terjadi proses pencocokan nilai jawaban yang diisikan oleh pemain dengan nilai jawaban yang dibangkitkan oleh sistem menggunakan algoritma *backtracking.* Untuk menjelaskan bagaimana alur proses penyelesaian jawaban pada kotak terisikan nilai tersebut, maka akan dijelaskan pada tabel 3.7.

Tabel 3.7 Proses pencocokan kedua nilai jawaban

Proses	Nilai jawaban diisi pemain	Nilai jawaban yang dibangkitkan pada Tabel 3.5	Pencocokan kedua nilai jawaban
1.	9 | 4 | 2 | 3	9 | 1 | 2 | 6	-
2.	9 | 4 | 2 | 3	9 | 1 | 2 | 6	9 | 4 | 2 | 3
3.	9 | 4 | 2 | 3	9 | 1 | 2 | 6	9 | 4 | 2 | 3
4.	9 | 4 | 2 | 3	9 | 1 | 2 | 6	9 | 4 | 2 | 3

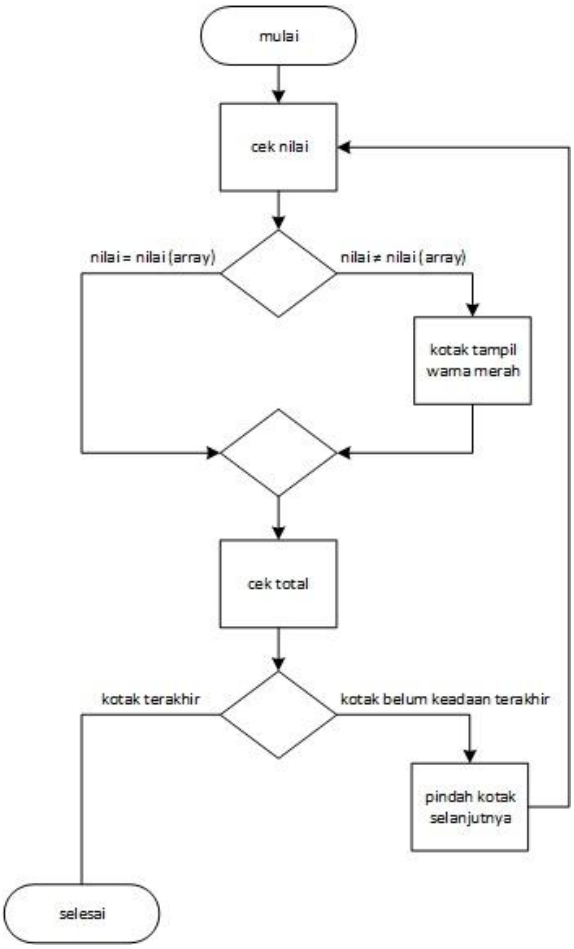
Dari proses tabel 3.7 terdapat proses-proses dalam pencocokan nilai jawaban dari nilai jawaban yang diisi oleh pemain dengan nilai jawaban yang dihasilkan oleh

sistem menggunakan algoritma *backtracking.* Keadaan awal, nilai jawaban yang telah diisikan pemain sudah terisikan nilai jawaban terlebih dahulu, baik dalam satu lajur maupun pada seluruh kotak jawaban. Selanjutnya, mencari nilai jawaban yang dibangkitkan oleh sistem menggunakan algoritma *backtracking* sebagai nilai pembanding dengan nilai jawaban yang telah diisi oleh pemain. Setelah kedua nilai jawaban didapat, maka proses selanjutnya adalah pencocokan nilai jawaban yang akan dijelaskan pada alur kerja bagaimana terjadinya proses pencocokan kedua nilai jawaban pada tabel tabel 3.8.

Tabel 3.8 Alur kerja pencocokan kedua nilai jawaban

Kotak jawaban pertama pada satu lajur	
Cek nilai	1. Apabila nilai jawaban yang diisi oleh pemain bernilai sama dengan nilai jawaban yang dibangkitkan oleh sistem didalam *array* menggunakan algoritma *backtracking,* maka sistem tidak akan menampilkan warna merah pada kotak jawaban karena sesuai dengan nilai yang dibangkitkan sistem. 2. Apabila nilai jawaban yang diisi oleh pemain bernilai tidak sama dengan nilai jawaban yang dibangkitkan oleh sistem didalam *array* menggunakan algoritma *backtracking,* maka sistem akan menampilkan warna merah pada kotak jawaban yang tidak sesuai dengan nilai yang dibangkitkan sistem.
Kotak jawaban kedua hingga terakhir pada satu lajur	
Cek nilai	

Dari alur kerja pencocokan kedua nilai jawaban tersebut, didapatlah hasil dari pencocokan kedua nilai jawaban. Dimana sistem akan menampilkan kotak berwana merah pada kotak jawaban yang tidak bernilai sama dengan nilai jawaban yang telah dibangkitkan oleh sistem. Alur kerja dalam pencocokan kedua nilai jawaban diatas akan digambarkan pada alur *flowchart* berikut :



Gambar 3.17 Proses pencocokan nilai jawaban

3.5 Analisis Kebutuhan Perangkat Lunak

Perangkat lunak (*software*) merupakan hal terpenting dalam mendukung kinerja sebuah sistem. Perangkat lunak digunakan dalam sebuah sistem untuk memberikan perangkat keras (*hardware*) sebuah perintah agar perangkat keras melakukan apa yang diinginkan oleh pemain. Berikut ini, kebutuhan perangkat lunak untuk menerapkan algoritma *backtracking* pada *puzzle* kakuro pada tabel 3.9.

Tabel 3.9 Spesifikasi perangkat lunak

Spesifikasi Perangkat Lunak
Windows XP, Windows 7, Windows 8
Greenfoot

3.5.1 Analisis Kebutuhan Perangkat Keras

Agar aplikasi dapat berjalan dengan baik, maka dibutuhkan perangkat keras (*hardware*) yang sesuai dengan kebutuhan aplikasi. Tabel 3.10 adalah spesifikasi minimum perangkat keras yang dibutuhkan agar dapat menjalankan aplikasi permainan *puzzle* kakuro secara optimal.

Tabel 3.10 Spesifikasi perangkat keras

Komputer	Spesifikasi Perangkat Keras
Prosesor	Prosesor 1,6 Ghz
Resolusi Layar	800 x 600 *pixel*
Memori	Memori 128 Mb
Hardisk	*Hardisk* 20 Gb
Keyboard dan *mouse*	Standar
Sound card	Standar

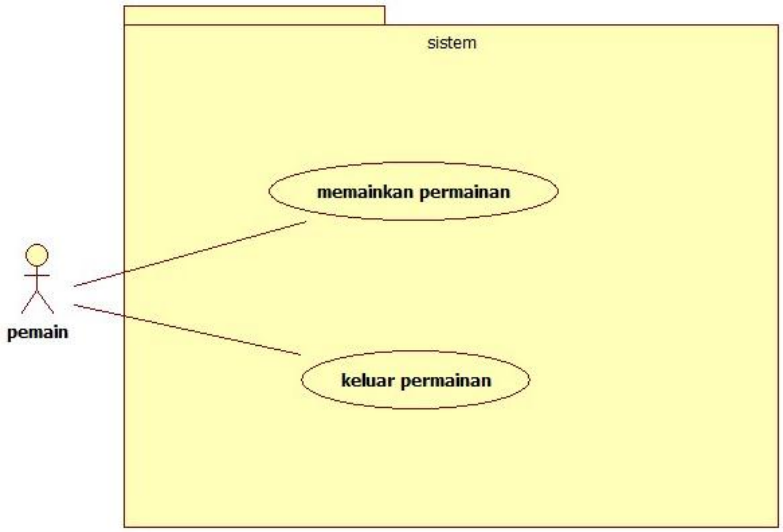
3.5.2 Analisis Kebutuhan Non Fungsional

Analisis kebutuhan non fungsional merupakan analisis yang dibutuhkan untuk menentukan spesifikasi kebutuhan sistem. Spesifikasi yang meliputi elemen atau komponen-komponen yang dibutuhkan untuk sistem yang akan dibangun hingga sistem tersebut diimplementasikan. Analisis kebutuhan ini menentukan spesifikasi masukan yang diperlukan sistem, keluaran yang akan dihasilkan sistem dan proses yang dibutuhkan untuk mengolah masukan sehingga menghasilkan suatu keluaran yang diinginkan.

3.5.3 Analisis Kebutuhan Fungsional

Pemodelan yang digunakan pada *game* yang akan dibuat adalah analisis perancangan dan pengembangan perangkat lunak berbasis objek, dimana alat yang akan digunakan untuk pengembangan perangkat lunak yaitu pemodelan *Unified Modeling Language* (UML).

3.5.3.1 *Use case Diagram*



Gambar 3.18 *Use case diagram*

Definisi *use case* berfungsi untuk menjelaskan fungsi dari *use case* yang terdapat pada diagram use case. Definisi *use case* dijelaskan pada tabel 3.11.

Tabel 3.11 Definisi *use case*

No	*Use Case*	Keterangan
1.	Memainkan Permainan	Proses untuk memulai memainkan permainan
2.	Keluar	Proses untuk keluar dari permainan

3.5.3.2 *Use Case Scenario*

Skenario *use case* merupakan penjelasan mengenai alur kejadian yang di dalamnya menjelaskan urutan interaksi antara aktor dengan *use case* tersebut. Adapun skenario *use case* digambarkan sebagai berikut.

Tabel 3.12 Tabel memainkan permainan

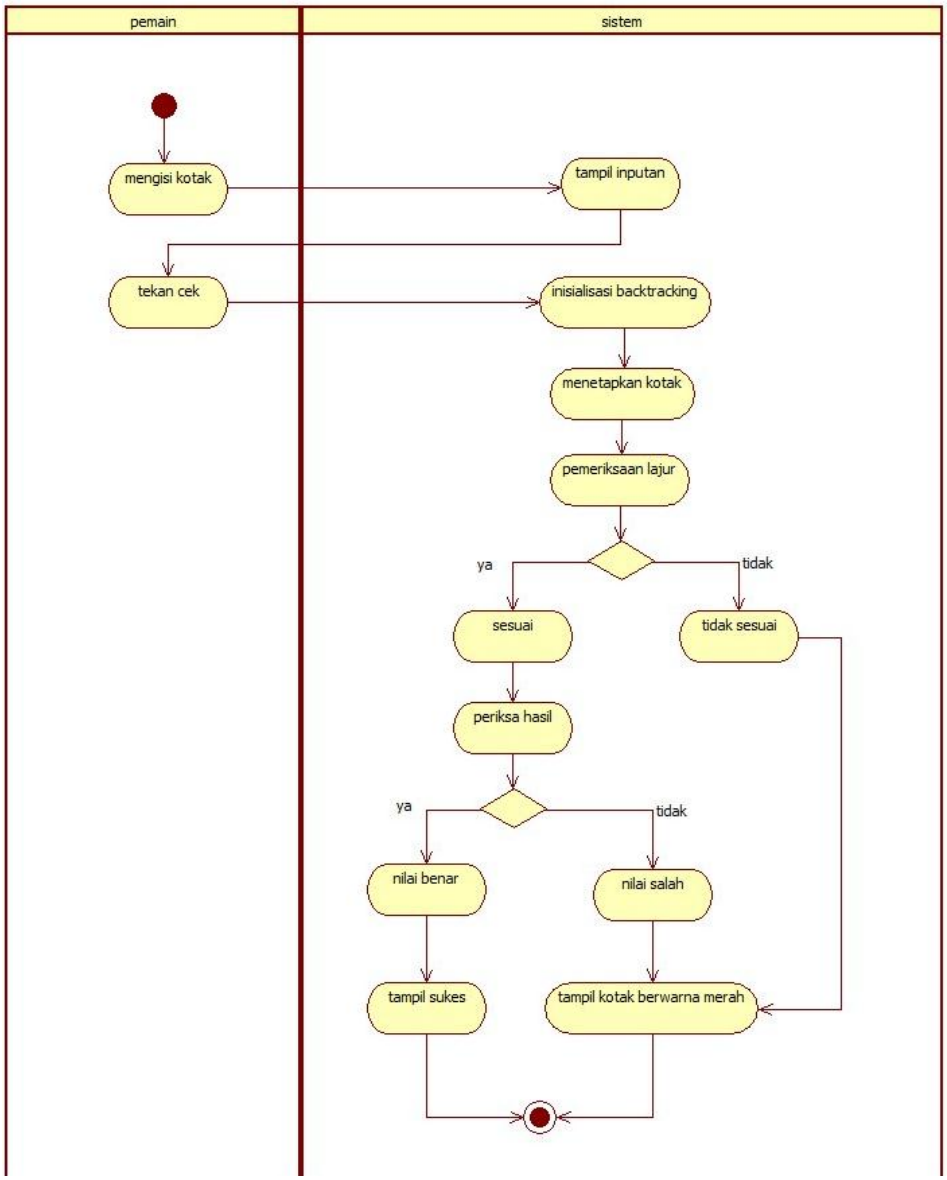
Identifikasi	
Nomor	1
Nama	Memainkan Permainan
Tujuan	Memainkan permainan

Deskripsi	Proses untuk memainkan permainan
Aktor	Pemain
Skenario Utama	
Kondisi Awal	Pemain berada di permainan
Aksi Aktor	Reaksi Sistem
	1. Mengatur blok permainan
	2. *Random* jumlah kolom
	3. *Random* jumlah baris
	4. Mengatur nilai pada blok dengan *random*
5. Mengecek nilai	
	6. Melakukan backtracking
	7. Cek jumlah kolom
	8. Jika blok pada kolom tidak kosong maka lakukan pencarian backtracking, jika kosong maka nilai input dinyatakan salah
	9. Cek jumlah baris
	10. Jika blok pada baris tidak kosong maka lakukan pencarian backtracking, jika kosong maka nilai input dinyatakan salah.
	11. Menampilkan sukses.
	12. Menampilkan salah.

3.5.3.3 *Activity Diagram*

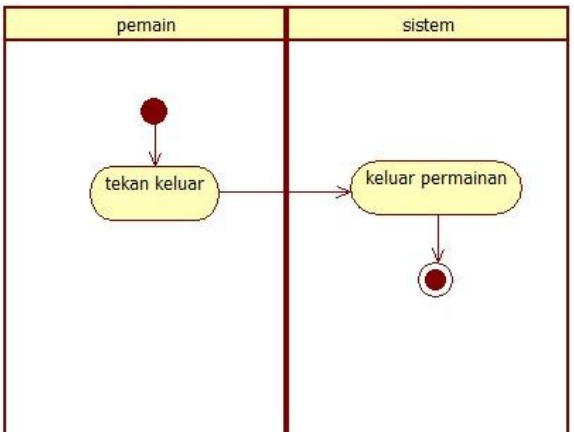
Activity diagram merupakan gambaran alur aktivitas pada permainan yang sedang dirancang, bagaimana masing-masing alur aktivitas berawal, *decision* yang mungkin terjadi, dan bagaimana berakhir. *Activity* diagram dijelaskan melalui gambar berikut.

1. *Activity* memainkan permainan



Gambar 3.19 *Activity* memainkan permainan

2. *Activity* keluar permainan

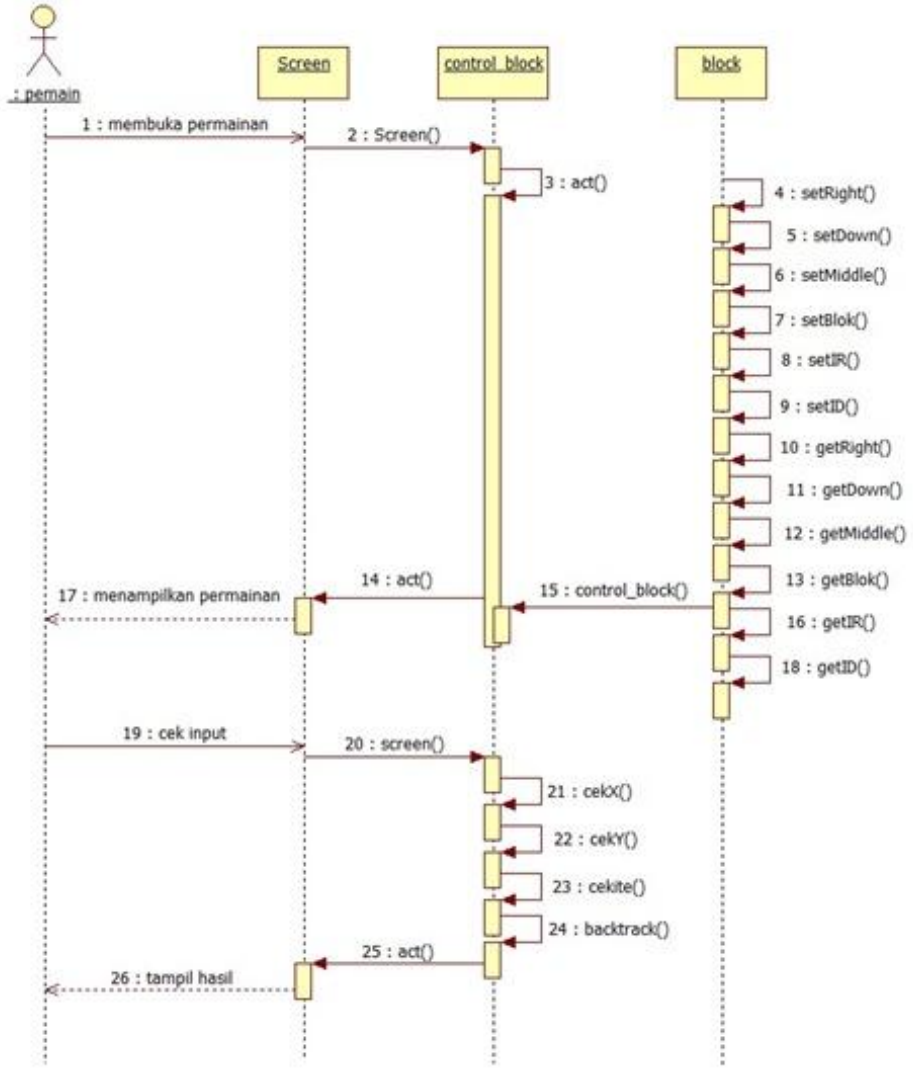


Gambar 3.20 *Activity* keluar permainan

3.5.3.4 *Squence Diagram*

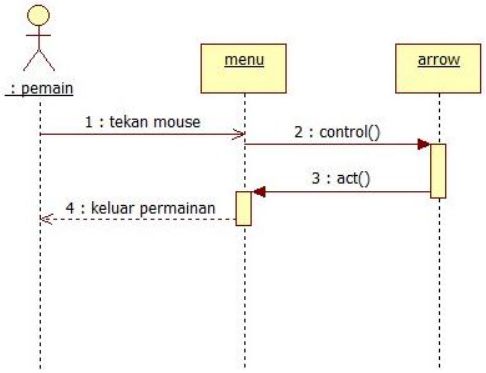
Sequence diagram menggambarkan tindakan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. *Sequence diagram* dapat dilihat pada gambar berikut.

1. *Squence diagram* memainkan permainan



Gambar 3.21 *Squence diagram* memainkan permainan

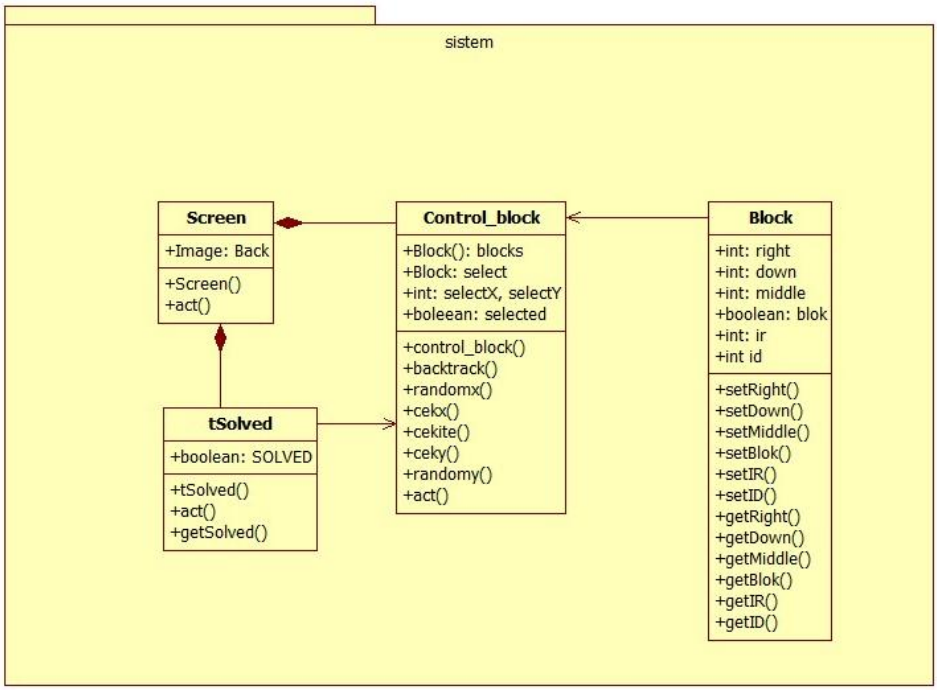
2. *Squence diagram* keluar permainan



Gambar 3.22 *Squence diagram* keluar permainan

3.5.3.5 *Class Diagram*

Class diagram merupakan gambaran sistem berupa relasi antar kelas-kelas ataupun paket-paketnya. *Class diagram* dijelaskan melalui gambar berikut.

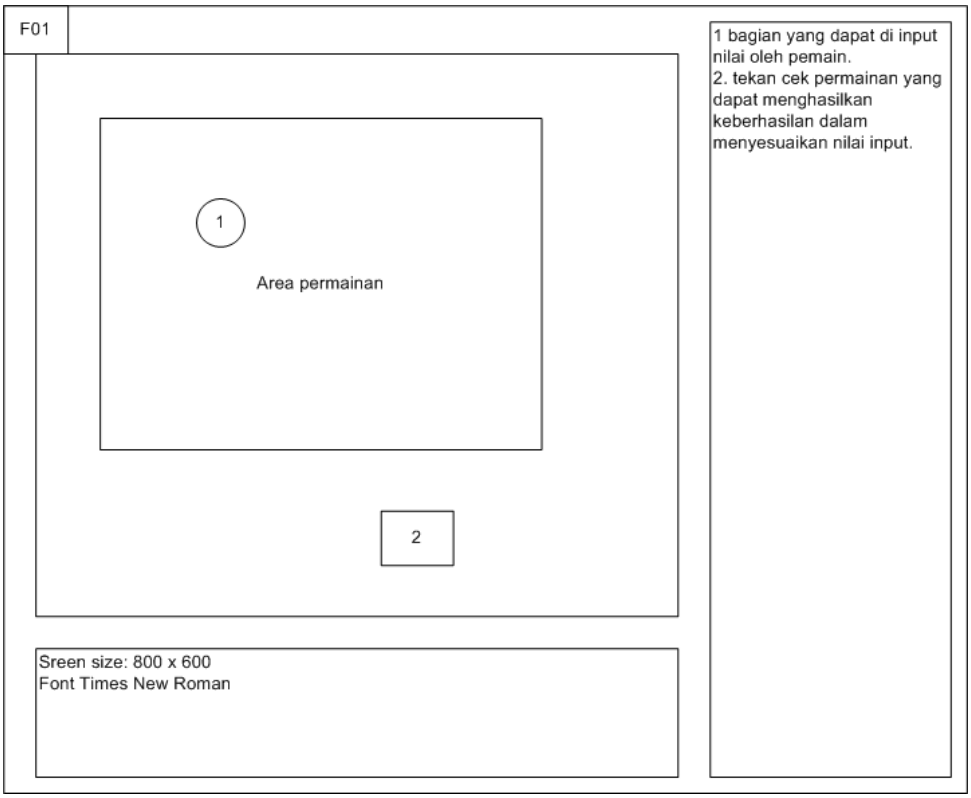


Gambar 3.23 *Class diagram* permainan *puzzle* kakuro

3.5.3.6 Perancangan Antarmuka

Perancangan antarmuka bertujuan untuk memberikan gambaran tentang aplikasi yang akan dibangun, sehingga akan mempermudah dalam mengimplementasikan aplikasi serta akan memudahkan pembuatan aplikasi. Antarmuka permainan merupakan halaman yang muncul pertama kali pada saat memainkan game. Berikut perancangan antarmuka permainan :

1. Perancangan antarmuka permainan.



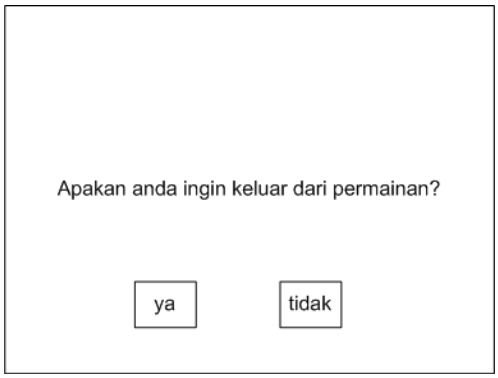
Gambar 3.24 Antarmuka permainan

2. Pesan ketika permaianan terselesaikan.



Gambar 3.25 Pesan ketika permaianan terselesaikan

3. Pesan keluar permainan.

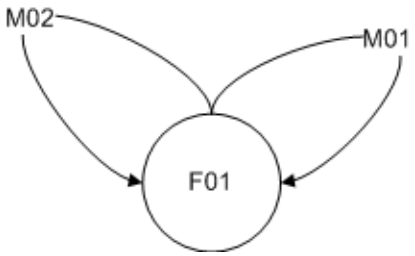


Gambar 3.26 Pesan keluar permainan

3.5.3.7 Jaringan Semantik

Untuk mengetahui keterhubungan dari setiap tampilan yang dibuat dapat menggunakan jaringan semantik.

adalah jaringan semantik pada *game* kakuro.



Gambar 3.27 Jaringan semantic