

Pemecah Sudoku Interaktif Berbasis Python

Proposal Tugas Akhir

Kelas TA 1

Muhammad Zakaria Musa

NIM: 1103130047



Program Studi Sarjana Teknik Informatika

Fakultas Informatika

Universitas Telkom

Bandung

2017

Lembar Persetujuan

Pemecah Sudoku Interaktif Berbasis Python

A Python-Based Interactive Sudoku Solver.

Muhammad Zakaria Musa

NIM: 1103130047

Proposal ini diajukan sebagai usulan pembuatan tugas akhir pada
Program Studi Sarjana Teknik Informatika
Fakultas Informatika Universitas Telkom

Bandung, 1 November 2017
Menyetujui

Calon Pembimbing 1

Calon Pembimbing 2

Yanti Rusmawati, Ph.D.
NIP: 15711785-1

Muhammad Arzaki, M.Kom.
NIP: 15871701-2

Abstrak

Sudoku adalah sebuah permainan teka-teki yang biasanya dimainkan oleh satu orang. Dalam pengerjaannya banyak pemain sudoku yang terjebak dan tidak dapat menyelesaikan teka-teki. Hal ini disebabkan oleh pemain salah memasukkan angka atau sudoku tidak memiliki solusi. Oleh karena itu perlu dibuat sebuah pemecah sudoku yang dapat memberikan keterpenuhan dari sebuah sudoku. Sehingga pemain dapat mengetahui sebuah sudoku memiliki solusi atau tidak. Dalam pengerjaannya metode yang digunakan adalah SAT *solver* yang dibuat dalam bahasa python. Setelah pembuatan aplikasi selesai akan dilakukan pengujian untuk mengontrol kualitas aplikasi. Pengujian akan menggunakan metode pengujian kotak hitam

Kata Kunci: metode formal, SAT solver, Sudoku, pengujian kotak hitam

Daftar Isi

Abstrak	i
Daftar Isi	ii
I Pendahuluan	1
1.1 Latar Belakang	1
1.2 Perumusan Masalah	2
1.3 Batasan Masalah	2
1.4 Tujuan	2
1.5 Rencana Kegiatan	2
1.6 Jadwal Kegiatan	3
II Kajian Pustaka	4
2.1 PISA	4
2.2 Sudoku	4
2.2.1 Sejarah Sudoku	5
2.2.2 Penyelesaian Sudoku	5
2.2.2.1 Memeriksa Baris, Kolom, dan Blok	5
2.2.2.2 Menemukan <i>Define</i>	6
2.2.2.3 <i>Backtrack</i> Ketika Mengalami Kemacetan	8
2.3 Metode Formal	9
2.4 Pengujian Kotak Hitam	9
2.5 Logika Proposisi	9
2.6 <i>Satisfiability Problem</i> (SAT)	10
2.6.1 Penyelesaian Sudoku Dengan SAT	10
2.7 <i>Waterfall Design</i>	11
III Metodologi	12
3.1 Analisa spesifikasi dan Kebutuhan sistem	12
3.2 Translasi spesifikasi sistem	12
3.3 Pembuatan aplikasi	13
3.3.1 Spesifikasi Aplikasi	13
3.3.2 <i>Analysis</i>	13
3.3.3 <i>Design</i>	13

3.3.4	<i>Coding</i>	14
3.4	Pengujian aplikasi	14
3.5	Analisis sistem	14
Daftar Pustaka		15
Lampiran		17

Bab I

Pendahuluan

1.1 Latar Belakang

Pada Tahun 2015 *The Organization for Economic Co-operation and Development* (OECD) merilis nilai dari *Programme for International Student Assessment* (PISA) [1]. Yang merupakan nilai kemampuan siswa pada matematika, membaca, dan membaca. Pada kategori matematika Indonesia menempati peringkat 65 dari 71 negara dengan nilai 386. Sedangkan nilai rata-rata OECD untuk matematika adalah 470 [2]. Hal ini sungguh memprihatinkan. Oleh karena itu diperlukan katalis yang dapat mengembangkan kemampuan *logical thinking* serta *problem solving* anak, salah satunya adalah dengan sudoku.

Sudoku berasal dari kata *Sūji wa dokushin ni kagiru* yang berarti angkanya harus tunggal [3] adalah suatu *puzzle* (teka-teki) yang direpresentasikan oleh sebuah matriks (*array* dua dimensi) berukuran $n^2 \times n^2$ yang dibangun dari n^2 dengan submatriks (atau blok) yang berukuran $n \times n$. Sudoku merupakan *puzzle* yang biasanya dimainkan oleh satu orang. Pada awal permainan, terdapat beberapa sel yang telah terisi yang disebut dengan pemberian (*givens*). Untuk menyelesaikan permainan ini, seorang pemain harus mengisi setiap sel yang belum terisi dengan angka di antara 1 sampai n^2 sedemikian sehingga setiap baris, setiap kolom, dan setiap blok (submatriks berukuran $n \times n$) memuat tepat satu bilangan di antara 1 sampai n^2 . Biasanya suatu sudoku didesain agar tepat memiliki satu kemungkinan solusi. Hal ini juga mengakibatkan sudoku dapat diselesaikan hanya dengan mengandalkan penalaran yang sederhana. Pengisian suatu sel dapat dilakukan dengan meninjau kemungkinan dari isi sebuah sel.

Banyak pemain yang terjebak pada teka-teki sudoku dan tidak dapat melanjutkan permainan. Hal ini terjadi karena pemain mengisi nilai yang salah atau sudoku tidak memiliki solusi. Oleh karena itu banyak pemain yang membutuhkan pemecah sudoku yang dapat memberikan keterpenuhan dari sebuah sudoku. Sehingga pemain dapat mengetahui sebuah sudoku memiliki solusi atau tidak.

Hingga saat ini, sudah banyak penelitian yang membahas penyelesaian sudoku secara matematis maupun komputasional. Salah satu metode yang cu-

kup dikenal adalah penyelesaian sudoku dengan memanfaatkan masalah keterpenuhan formula proposisional (*propositional satisfiability problem*). Dengan pendekatan ini, syarat-syarat yang harus dipenuhi oleh suatu sudoku dimodelkan dengan satu atau lebih formula logika proposisi [4, 5]. Keterpenuhan (*satisfiability*) dari himpunan formula yang memodelkan syarat-syarat ini akan menjamin bahwa suatu sudoku memiliki suatu solusi.

Pada tugas akhir ini, penulis akan membuat pemecah sudoku interaktif berbasis python. Bahasa python dipilih karena implementasi SAT solver dalam bahasa python masih sedikit dibandingkan pada bahasa C dan C++ [6]. Aplikasi akan menggunakan pengujian kotak hitam untuk menilai kualitas aplikasi [7].

1.2 Perumusan Masalah

Berdasarkan latar belakang tersebut, rumusan masalah pada tugas akhir ini adalah bagaimana cara membuat SAT solver berbasis python untuk pemecah sudoku yang interaktif.

1.3 Batasan Masalah

Batasan pada tugas akhir ini terbatas pada:

1. Penulis hanya membangun aplikasi sudoku solver berukuran (4×4) , (9×9) , (16×16) .
2. Untuk setiap ukuran sudoku memiliki tiga tingkat kesulitan. Yaitu mudah, menengah, sulit.
3. Program mengeluarkan keterpenuhan dari sudoku.
4. Program mengeluarkan solusi dari sudoku jika pengguna menginginkan.
5. Aplikasi akan diuji dengan metode pengujian kotak hitam (*black box testing*).

1.4 Tujuan

Tujuan yang ingin dicapai pada tugas akhir ini adalah untuk membuat pemecah sudoku yang interaktif.

1.5 Rencana Kegiatan

Pada pengerjaan tugas akhir ini beberapa hal yang akan saya lakukan adalah sebagai berikut:

1. Analisa spesifikasi dan Kebutuhan sistem.
2. Translasi spesifikasi sistem.

3. Pembuatan aplikasi.
4. Pengujian aplikasi.
5. Analisis sistem.
6. Penulisan laporan.

1.6 Jadwal Kegiatan

Jadwal pengerjaan tugas akhir sesuai dengan alur yang telah dibuat.

Tabel 1.1: Jadwal Kegiatan.

No	Jenis Kegiatan	Bulan						
		November 2017	Desember 2017	Januari 2018	Februari 2018	Maret 2018	April 2018	Mei 2018
1	Analisa spesifikasi dan kebutuhan sistem							
2	Translasi spesifikasi sistem							
3	Pembuatan aplikasi							
4	Pengujian aplikasi							
5	Analisis sistem							
6	Penulisan Laporan							

Bab II

Kajian Pustaka

Pada bab ini penulis akan menjelaskan teori yang digunakan selama pengerjaan tugas akhir.

2.1 PISA

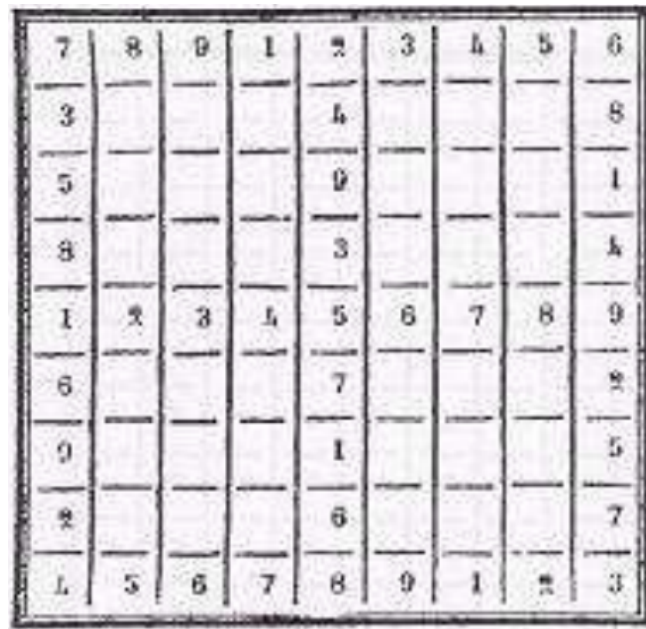
Programme for International Student Assessment adalah sebuah program yang dilakukan tiga tahun sekali oleh *The Organization for Economic Co-operation and Development* (OECD) dengan tujuan untuk membuat data yang dapat dibandingkan yang akan memungkinkan negara-negara memperbaiki kebijakan dan hasil pendidikan mereka [1]. Siswa yang dijadikan *sample* pada PISA berumur 15 tahun, hal ini dilakukan karena mereka berada pada akhir pendidikan wajib. Hal yang diujikan dalam PISA berupa matematika, sains, dan membaca.

2.2 Sudoku

Sudoku berasal dari kata *Sūji wa dokushin ni kagiru* yang berarti angkanya harus tunggal [3]. Sudoku dipopulerkan di Jepang oleh *Nikoli in the paper Monthly Nikolist* pada April 1984. Sudoku adalah suatu *puzzle* (teka-teki) yang direpresentasikan oleh sebuah matriks (*array* dua dimensi) berukuran $n^2 \times n^2$ yang dibangun dari n^2 dengan submatriks (atau blok) yang berukuran $n \times n$. Sudoku merupakan *puzzle* yang biasanya dimainkan oleh satu orang. Pada awal permainan, terdapat beberapa sel yang telah terisi yang disebut dengan pemberian (*givens*). Untuk menyelesaikan permainan ini, seorang pemain harus mengisi setiap sel yang belum terisi dengan angka di antara 1 sampai n^2 sedemikian sehingga setiap baris, setiap kolom, dan setiap blok (submatriks berukuran $n \times n$) memuat tepat satu bilangan di antara 1 sampai n^2 . Biasanya suatu sudoku didesain agar tepat memiliki satu kemungkinan solusi. Hal ini juga mengakibatkan sudoku dapat diselesaikan hanya dengan mengandalkan penalaran yang sederhana. Pengisian suatu sel dapat dilakukan dengan meninjau kemungkinan dari isi sebuah sel.

2.2.1 Sejarah Sudoku

Sebelum sudoku modern berkembang. *Puzzle latin square* terlebih dahulu berkembang yang di repretasikan oleh sebuah matriks berukuran $n \times n$ yang memiliki angka *1 hingga* yang harus diisi oleh pemain. untuk menyelesaikan *puzzle* setiap baris dan kolom pada *latin square* harus memiliki nilai angka yang berbeda. *Latin square* pertama kali diciptakan oleh Euler pada tahun 1783 [8]. Pada 19 November 1892 *Le Siècle* sebuah surat kabar di perancis mempublikasikan *magic square* yaitu sebuah teka-teki yang di repretasikan oleh sebuah matriks berukuran 9×9 yang memiliki dua buah submatriks berukuran 3×3 . Tujuan dari *magic square* adalah pemain mengisi baris dan kolom sehingga nilai penjumlahan setiap baris dan kolom itu sama.



7	8	9	1	2	3	4	5	6
3				4				8
5				9				1
8				3				4
1	2	3	4	5	6	7	8	9
6				7				2
9				1				5
2				6				7
4	5	6	7	8	9	1	2	3

Gambar 2.1: Benduk *magic square*. (gambar diambil dari [9])

Sudoku modern sendiri lahir dari Howard Garns yang dipublikasikan di *Dell Magazines* pada 1979 [10].

2.2.2 Penyelesaian Sudoku

Dalam menyelesaikan sudoku terdapat banyak cara. Namun salah satu cara yang paling efektif adalah dengan berikut:

2.2.2.1 Memeriksa Baris, Kolom, dan Blok

Pada sudoku dengan ukuran 9×9 Untuk menyelesaikan sudoku pemain tidak boleh menaruh angka yang sama di baris, kolom, atau blok 3×3 .

	1	9			6			
2		8	3	1		5		6
	6			7			1	
	3	7	6		1	7		4
1	8	7	2		4		5	3
		4	8					9
4		1		6			3	
			1			2		5
	9			2	8	4	6	

Gambar 2.2: 7 tidak mungkin pada sel tersebut. (gambar diambil dari [?])

2.2.2.2 Menemukan *Define*

Define adalah angka yang pasti ada pada sel itu. *Define* dipengaruhi oleh *given* suatu sudoku. Untuk mempermudah menemukan *define* pemain memulai mencarinya dari angka 1 lalu menarik sebuah garis dari setiap *given* bernilai 1. Ketika hanya ada satu sel tersisa pada sebuah blok maka sel itu memiliki *Define* angka 1.

	1	9			6			
2		8	3	1		5		6
	6			7			1	
	3	7	6		1	7		4
1	8	7	2		4		5	3
		4	8			1		9
4		1		6			3	
			1			2		5
	9			2	8	4	6	1

Gambar 2.3: Menemukan *define*. (gambar diambil dari [?])

Ulangi proses yang sama hingga angka 9.

	1	9			6			2
2	4	8	3	1		5		6
	6			7			1	
9	3	2	6		1	7	8	4
1	8	7	2	9	4	6	5	3
6		4	8		7	1	2	9
4	2	1		6			3	
8		6	1			2		5
	9			2	8	4	6	1

Gambar 2.4: Semua *define* ditemukan. (gambar diambil dari [?])

2.2.2.3 *Backtrack* Ketika Mengalami Kemacetan

Ketika pemain mengalami kemacetan dalam mengerjakan sudoku. Hal ini terjadi karena pemain salah memasukkan angka pada sel diluar *define*. Salah satu caranya adalah dengan memberi catatan untuk setiap blok dengan angka yang mungkin berada pada sel tersebut.

	1	9			6			2
2	4	8	3	1		5		6
	6			7			1	
9	3	2	6		1	7	8	4
1	8	7	2	9	4	6	5	3
6		4	8		7	1	2	9
4	2	1		6			3	
8	5, 3, 7	6	1			2		5
5, 3, 7	9	5, 3, 7		2	8	4	6	1

Gambar 2.5: Catatan yang berisi setiap kemungkinan angka. (gambar diambil dari [?])

2.3 Metode Formal

Metode formal adalah teknik yang digunakan untuk memodelkan suatu sistem yang kompleks [11]. Metode formal dapat digunakan untuk mengembangkan sebuah aplikasi baik itu perangkat lunak atau perangkat keras. Pada masa spesifikasi, metode formal dapat digunakan untuk memberikan gambaran dari sistem yang akan dikembangkan sedetail yang diinginkan. Metode formal dapat digunakan untuk memandu kegiatan pengembangan selanjutnya dan dapat digunakan untuk memverifikasi bahwa pra-syarat untuk sistem yang dikembangkan telah lengkap dan dapat dilanjutkan ke tingkat selanjutnya.

2.4 Pengujian Kotak Hitam

Pengujian kotak hitam adalah teknik pengujian yang tidak memerlukan pengetahuan kode dari aplikasi [7]. Penguji biasanya melakukan pengujian kotak hitam melalui antarmuka pengguna dari aplikasi dengan diberikan *test case* masukan dan menganalisa keluaran dari aplikasi.

2.5 Logika Proposisi

Logika proposisi adalah bahasa formal untuk memodelkan situasi yang kita sehingga kita dapat memberi alasan tentang hal itu secara formal [11]. Logika

proposisi terdiri dari sebuah nilai kebenaran dari proposisinya.

Operator yang digunakan pada logika proposisi adalah sebagai berikut

- Konjungsi(dan) pada operator ini kedua proposisi yang dihubungkan harus bernilai benar agar formula bernilai *true* . Operator ini dilambangkan dengan \wedge .
- Disjungsi(atau) pada operator ini salah satu proposisi yang dihubungkan harus bernilai benar agar formula bernilai *true*. Operator ini dilambangkan dengan \vee
- Negasi(tidak) pada operator ini suatu proposisi akan memiliki nilai sebaliknya dari sebuah proposisi. Operator ini dilambangkan dengan \neg
- Implikasi(jika-maka) pada operator ini digunakan untuk merepresentasikan kata “jika p maka q” sebuah formula akan bernilai *false* jika nilai p benar dan nilai q salah. Operator ini dilambangkan dengan \rightarrow .
- Biimplikasi(jika dan hanya jika)pada operator ini suatu formula hanya akan bernilai *true* jika kedua proposisinya memiliki nilai kebenaran yang sama. Operator ini dilambangkan dengan \leftrightarrow

2.6 *Satisfiability Problem*(SAT)

Masalah SAT (*SAT problem*) adalah salah satu masalah penting dalam logika komputasional [11]. Dalam pengerjaannya SAT berfokus pada membuktikan sebuah klausa *satisfiable* sehingga SAT berfokus pada menemukan sebuah model yang membuat klausa tersebut *satisfiable*. Jika tidak ditemukan sebuah model yang membuat klausa tersebut *satisfiable* maka klausa tersebut *unsatisfiable*. Masalah SAT merupakan masalah NP-complete, hingga saat ini tidak terdapat algoritma yang efisien untuk memecahkan masalah tersebut.

2.6.1 Penyelesaian Sudoku Dengan SAT

Sudoku memiliki beberapa aturan contohnya untuk sudoku berukuran 9×9 yaitu :

1. Setiap baris memuat bilangan antara 1 hingga 9.
2. Setiap kolom memuat bilangan antara 1 hingga 9.
3. Setiap submatriks atau blok 3×3 memuat bilangan antara 1 hingga 9.
4. Setiap sel memuat tepat satu bilangan antara 1 hingga 9.

Dari aturan-aturan tersebut akan ditranslasikan menjadi bentuk CNF lalu digunakan pada SAT *solver*. Dengan aturan yang telah ditranslasikan dalam CNF sebagai berikut:

1. Setiap baris memuat bilangan antara 1 hingga 9 :

$$\bigwedge_{y=1}^9 \bigwedge_{z=1}^9 \bigvee_{x=1}^9 s_{xyz}$$

2. Setiap kolom memuat bilangan antara 1 hingga 9 :

$$\bigwedge_{x=1}^9 \bigwedge_{z=1}^9 \bigvee_{y=1}^9 s_{xyz}$$

3. Setiap submatriks atau blok 3×3 memuat bilangan antara 1 hingga 9 :

$$\bigvee_{z=1}^9 \bigwedge_{i=0}^2 \bigwedge_{j=0}^2 \bigwedge_{x=1}^3 \bigwedge_{y=1}^3 s_{(3i+x)(3j+y)z}.$$

4. Setiap sel memuat tepat satu bilangan antara 1 hingga 9 :

$$\bigwedge_{x=1}^9 \bigwedge_{y=1}^9 \bigwedge_{z=1}^8 \bigwedge_{i=z+1}^9 (\neg s_{xyz} \vee \neg s_{xyi})$$

2.7 Waterfall Design

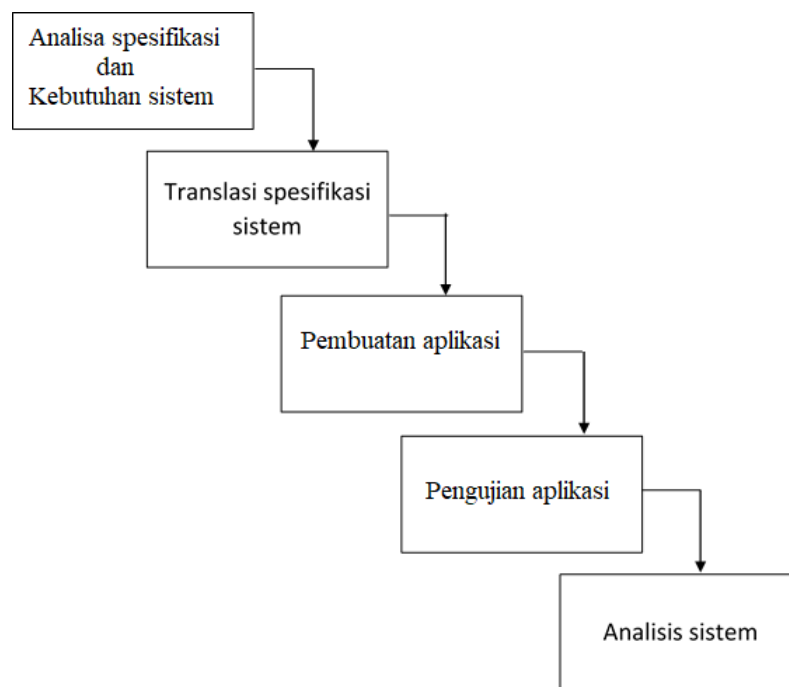
Waterfall atau air terjun adalah model yang diperkenalkan oleh Dr. Winston W. Royce pada tahun 1970. Model ini menuntun pengembang untuk melakukan pengembangan perangkat lunak yang akan dibangun secara sistematis dan mempunyai keterkaitan antara prosesnya. dalam arti tahapan proses berikutnya tidak akan dapat di wujudkan jika tahapan sebelumnya tidak terselasaikan.

Ada dua langkah penting yang harus ada dalam pembuatan perangkat lunak. pertama ialah analisis lalu di ikuti oleh *coding*, jika di bagi akan menjadi beberapa tahap yang spesifik yaitu *System Requirements* \rightarrow *Software Requirements* \rightarrow *Analysis* \rightarrow *Program Design* \rightarrow *Coding* \rightarrow *Testing* [12].

Bab III

Metodologi

Pada bab ini penulis menjelaskan mengenai metodologi yang digunakan selama pengerjaan tugas akhir.



Gambar 3.1: Metodologi.

3.1 Analisa spesifikasi dan Kebutuhan sistem

Menganalisa spesifikasi-spesifikasi pada teka-teki sudoku, lalu membuat kebutuhan dari aplikasi sistem yang akan mengeluarkan *use case* aplikasi.

3.2 Translasi spesifikasi sistem

Spesifikasi yang sudah di dapat diubah dalam bentuk logika proposisi lalu dijadikan dalam bentuk CNF.

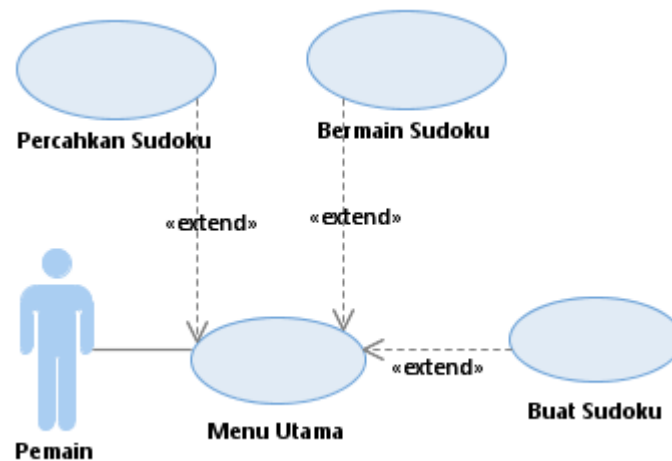
3.3 Pembuatan aplikasi

Spesifikasi yang sudah dalam bentuk CNF akan digunakan pada *SAT Solver*. Lalu *use case* yang didapat akan dibuat fungsionalitasnya. Pembuatan aplikasi menggunakan *waterfall design* tahapannya berupa:

3.3.1 Spesifikasi Aplikasi

Spesifikasi aplikasi adalah sebagai berikut:

1. Bermain sudoku.
2. Membuat sudoku.
3. Memecahkan sudoku.



Gambar 3.2: *Use Case Diagram*.

3.3.2 *Analysis*

Dalam *analysis*, penulis menjabarkan mulai dari hal mendasar yaitu kendala global dalam pengembangan perangkat lunak yang sudah ada. Adapun masalahnya adalah minimnya aplikasi pemecah sudoku yang interaktif.

3.3.3 *Design*

Fitur yang akan ada pada aplikasi tersebut adalah sebagai berikut

1. Bermain sudoku.
2. Membuat sudoku.
3. Pecahkan sudoku.

3.3.4 *Coding*

Lalu dalam tahap *coding* penulis akan menggunakan bahasa pemrograman python dengan SAT *solver* [13] sebagai dasarnya. Setelah itu akan ditambahkan GUI dan *use case* yang telah di buat.

3.4 Pengujian aplikasi

Pada tahap ini aplikasi akan diujikan kebeberapa penguji dengan diberikan lembar *test case* yang meliputi beberapa aspek kualitas pada aplikasi. Teknik yang digunakan pada pengujian adalah pengujian kotak hitam [7].

3.5 Analisis sistem

Menganalisa kelayakan kualitas dari aplikasi berdasarkan hasil dari pengujian.

Daftar Pustaka

- [1] “Cara memecahkan teka teki sudoku,” <https://id.wikihow.com/Memecahkan-Teka-teki-Sudoku>.
- [2] “The latest ranking of top countries in math, reading, and science is out,” <http://www.businessinsider.sg/pisa-worldwide-ranking-of-math-science-reading-skills-2016-12/?r=US&IR=T>.
- [3] “Ed pegg jr.’s math games: Sudoku variations,” <http://www.mathpuzzle.com/MAA/41-Sudoku>
- [4] G. Kwon and H. Jain, “Optimized CNF encoding for sudoku puzzles,” in *Proc. 13th International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR2006)*, 2006, pp. 1–5.
- [5] I. Lynce and J. Ouaknine, “Sudoku as a SAT Problem.” in *ISAIM*, 2006.
- [6] “Understanding SAT by Implementing a Simple SAT Solver in Python,” <http://sahandsaba.com/understanding-sat-by-implementing-a-simple-sat-solver-in-python.html>.
- [7] B. Beizer, *Black-box testing: techniques for functional testing of software and systems*. John Wiley & Sons, Inc., 1995.
- [8] S. Jones, P. Roach, and S. Perkins, “Properties of sudoku puzzles,” in *Proceedings of the 9th international conference on Software Engineering*. IEEE Computer Society Press, 11 2007, pp. 7–11.
- [9] “Supplément de l’article: Les ancêtres français du sudoku,” <https://web.archive.org/web/20061210103525/http://cboyer.club.fr/multimagic/SupplAnc>
- [10] “The answer men,” <http://content.time.com/time/magazine/article/0,9171,2137423,00.htm>
- [11] M. Huth and M. Ryan, *Logic in Computer Science: Modelling and reasoning about systems*. Cambridge University Press, 2004.
- [12] W. W. Royce, “Managing the development of large software systems: concepts and techniques,” in *Proceedings of the 9th international conference*

on Software Engineering. IEEE Computer Society Press, 1987, pp. 328–338.

- [13] “A simple SAT solver implemented in Python (with some non-SAT abilities).” <https://github.com/stephenroller/satsolver>.

Lampiran

Pada bab ini akan berisi lampiran dari pengerjaan tugas akhir, contohnya skrip NuSMV.