Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 3, No. 8, Agustus 2019, hlm. 7642-7647 http://j-ptiik.ub.ac.id

Pembangunan Kakas Bantu Perhitungan Kualitas Kode (*Quality Rate*) menggunakan Metrik Perangkat Lunak

Odhia Yustika Putri[1], Bayu Priyambadha[2], Arief Andy Soebroto[3]

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]odhiayustika@student.ub.ac.id, [2]bayu_priyambadha@ub.ac.id, [3]ariefas@ub.ac.id

Abstrak

Perangkat lunak yang berkualitas adalah perangkat lunak yang tidak memiliki defisiensi kode. Sebelumnya Beranic dkk pada tahun 2018 melakukan penelitian berjudul "*Towards a Reliable Identification of Deficient Code with a Combination of Software Metrics*" mengenai penggunaan kombinasi 8 (delapan) jenis metrik perangkat lunak berorientasi objek untuk mendeteksi defisiensi kode pada perangkat lunak tersebut. Pendeteksian defisiensi kode ini dapat digunakan pada perhitungan kualitas kode (*quality rate*) yang dapat menjadi acuan baik buruknya perangkat lunak tersebut. Pada penelitian tersebut perhitungan metrik dilakukan dengan menggunakan kakas bantu terpisah untuk menghitung kualitas kode, sehingga membutuhkan waktu dan sumber daya yang banyak. Oleh karena itu, pada penelitian ini dibangunlah kakas bantu perhitungan kualitas kode menggunakan metrik perangkat lunak. Pembangunan kakas bantu dilakukan menggunakan metode siklus hidup perangkat lunak Waterfall dan dibangun dalam bahasa pemrograman Java. Kakas bantu dapat berjalan pada platform desktop dan menggunakan berkas proyek Java sebagai masukannya. Kakas bantu ini menghasilkan 3 kebutuhan fungsional dan 1 kebutuhan non-fungsional. Perancangan kakas bantu ini dilakukan dengan berorientasi objek dan menghasilkan 1 *class diagram*, 3 *sequence diagram*, 5 rancangan algoritme, dan 3 rancangan antarmuka. Implementasi kakas bantu ini dilakukan sesuai rancangan menggunakan *library* JavaFX dan ASTParser. Hasil pengujian *white-box* dan *black-box* yang telah dilakukan menunjukkan bahwa kakas bantu ini memiliki persentase validasi sebesar 100%.

Kata kunci: kakas bantu, kualitas kode, metrik perangkat lunak

Abstract

A good quality software is usually understood as a lack of errors within the software. The previous study entitled "Towards a Reliable Identification of Deficient Code with a Combination of Software Metrics" has been done by Beranic in 2018 and showed the use of combination of 8 object-oriented software metrics to identify deficient code in certain software. The deficient code detection can be used to calculate the quality rate. However, the calculation of deficient code and quality rate are done separately. To overcome this problem, a tool is made for the calculation of quality rate with a combination of software metrics. This tool was built using Software Development Life Cycle (SDLC) Waterfall method and coded in Java. This tool runs on a desktop platform and use Java project as its input. Requirement engineering process in this tool produced 3 functional requirements and 1 non-functional requirement. The design is in object-oriented and produced 1 class diagram, 3 sequence diagrams, 5 algorithms, and 3 user interface design. The design is develop using JavaFX and ASTParser library. The outcome of the white-box and black-box tests indicate that this tool has a validation percentage of 100%.

Keywords: *tools, quality rate, software metrics*

1. PENDAHULUAN

Menyajikan produk yang berkualitas haruslah menjadi prioritas utama dalam proyek perangkat lunak. Kualitas perangkat lunak bergantung pada proses pengembangan perangkat lunak tersebut (Akbar, et. al., 2018), termasuk analisis pada kode. Kualitas perangkat lunak juga seringkali dikaitkan dengan tidak adanya kesalahan yang terdapat didalamnya

(Kan, 2002), dengan kata lain perangkat lunak yang berkualitas adalah perangkat lunak yang tidak memiliki defisiensi kode. Defisiensi kode dapat digunakan dalam perhitungan kualitas kode (*quality rate*) yang menjadi acuan baik buruknya perangkat lunak tersebut.

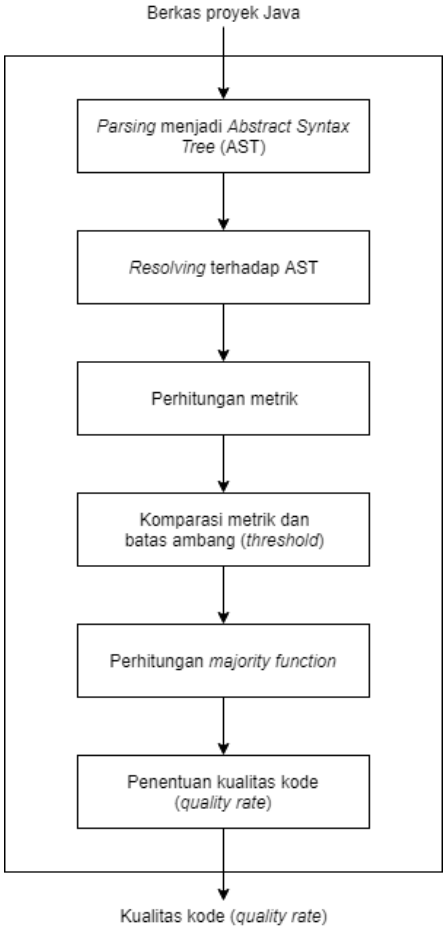
Terdapat beberapa strategi dalam mendeteksi defisiensi kode, seperti menggunakan metrik perangkat lunak. Namun penggunaan metrik individual hanya akan mengevaluasi satu dimensi kualitas saja yang biasanya berujung pada hasil yang *false positive* (Sharma, et. al., 2018). Sebagai tambahan, gambaran umum yang seimbang tentang kualitas perangkat lunak dapat diperoleh dengan mengkombinasikan beberapa metrik (Bouwers, et. al., 2012).

Meskipun terdapat banyak studi mengenai penggunaan kombinasi metrik perangkat lunak, namun sangat sulit untuk menentukan komposisi metrik yang dapat mengakomodir pendeteksian defisiensi kode. Oleh karena itu Beranic, et al. (2018) mengemukakan penelitian mengenai deteksi defisiensi kode menggunakan *majority function* pada tiap-tiap aspek kualitas pada metrik perangkat lunak. Identifikasi juga dilakukan menggunakan *threshold value* (ambang batas) sehingga evaluasi yang dilakukan akan semakin akurat. Aspek-aspek kualitas yang akan dievaluasi yaitu ukuran (*size*), kompleksitas (*complexity*), kopling (*coupling*), dan kohesi (*cohesion*). Sedangkan metrik yang digunakan untuk aspek ukuran adalah *Source Line of Code* (SLOC), *Average Method Size* (AMS), dan *Number of Method* (NOM); metrik yang digunakan untuk aspek kompleksitas adalah *Weighted Method per Class* (WMC), *Average Cyclomatic Complexity* (ACC), dan *Max Nesting* (NS); metrik yang digunakan untuk aspek kopling adalah *Coupling Between Object* (CBO); serta metrik yang digunakan untuk aspek kohesi adalah *Lack of Cohesion in Method* (LCOM).

Namun hingga saat ini perhitungan kualitas kode berdasarkan penelitian Beranic, et al. tersebut masih dilakukan dengan kakas bantu terpisah dan dilanjutkan dengan perhitungan manual sehingga memakan waktu apalagi jika dilakukan dalam proyek besar. Berdasarkan permasalahan-permasalahan tersebut di atas, dibuatlah suatu sistem berupa kakas bantu yang dapat melakukan perhitungan kualitas kode (*quality rate*) dengan metode di atas secara otomatis. Kakas bantu ini diharapkan dapat bermanfaat bagi para pengembang dalam menentukan kualitas dan memudahkan pemeliharaan perangkat lunak. Pembangunan dari kakas bantu ini akan dilakukan menggunakan bahasa pemrograman Java dan akan berjalan pada platform *desktop*.

2. METODE KALKULASI

Untuk mendapatkan hasil kualitas kode (quality rate), dilakukan kalkulasi yang terdiri dari beberapa tahapan. Tahap-tahap tersebut adalah melakukan *parsing* terhadap masukan berapa berkas proyek Java ke dalam *Abstract Syntax Tree* (AST), melakukan *resolving* terhadap AST, menghitung metrik perangkat lunak masing-masing kelas, melakukan komparasi hasil perhitungan metrik dan batas ambang (*threshold*), menghitung *majority function*, dan menentukan kualitas kode (*quality rate*). Tahapan-tahapan ini terdapat dalam Gambar 1.



Gambar 1. Diagram Alir Perhitungan Kualitas Kode

2.1 Batas Ambang Metrik Perangkat Lunak

Batas ambang untuk masing-masing metrik perangkat lunak dalam tiap kelas pada berkas proyek dalam bahasa pemrograman Java

terdapat dalam Tabel 1.

Tabel 1. Batas Ambang Metrik Perangkat Lunak pada Java

Metrik Perangkat Lunak		Nilai Ambang pada Java
Source Line of Code	SLOC	197
Number of Method	NOM	19
Average Method Size	AMS	51
Weighted Method per Class	WMC	33
Average Cyclomatic Complexity	ACC	3
Maximum Nesting	NS	3
Coupling Between Object	CBO	11
Lack of Cohesion in Methods	LCOM	71

2.2 *Majority Function*

Majority Function masing-masing aspek kualitas dapat dihitung menggunakan Persamaan 1,

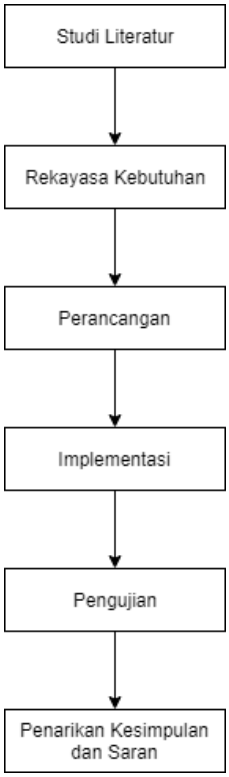
$$Majority_{y_j} = \frac{\sum_{i=0}^{n} x_i}{n} \quad (1)$$

dimana y_j adalah aspek kualitas yang ingin dihitung, x_i adalah hasil perbandingan metrik-metrik dalam aspek tersebut, dan n adalah jumlah metrik yang digunakan. Jika hasil perhitungan masing-masing metrik melampaui nilai ambangnya, nilai x_i adalah 1, sebaliknya nilai x_i adalah 0.

3. METODOLOGI PENELITIAN

Tipe penelitian yang digunakan dalam pembauatan skripsi Pembangunan Kakas Bantu Perhitungan Kualitas Kode (*Quality Rate*) Menggunakan Metrik Perangkat Lunak adalah tipe penelitian pengembangan (*development*). Tipe penelitian ini akan menghasilkan produk atau artefak sebagai solusi dalam penelitian.

Sedangkan metode penelitian yang digunakan dalam penelitian ini adalah metode *Waterfall* dimana masing-masing tahapan dilakukan secara berurutan dan sekuensial. Tahapan dalam penelitian ini terdiri atas tahap Studi Literatur, tahap Rekayasa Kebutuhan, tahap Perancangan, tahap Implementasi, tahap Pengujian, dan tahap Penarikan Kesimpulan dan Saran, yang terdapat dalam Gambar 2.



Gambar 2. Metodologi Penelitian

2.2 Studi Literatur

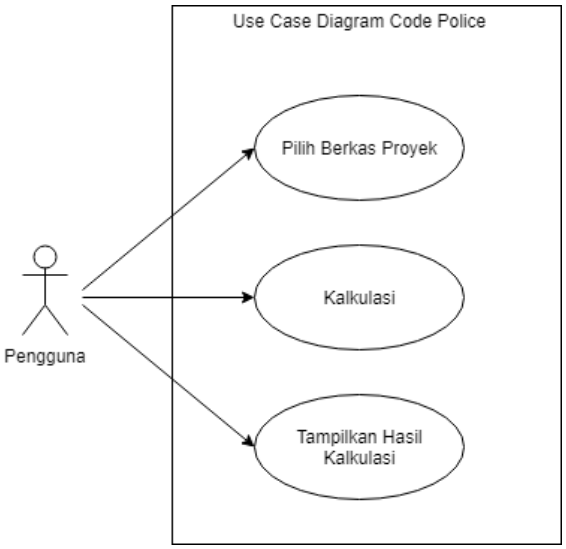
Dasar teori mengenai pengerjaan penelitian Pembangunan Kakas Bantu Perhitungan Kualitas Kode Menggunakan Metrik Perangkat Lunak ini diperoleh dari artikel, buku, jurnal, konferensi, dan penelitian terkait *Software Development Life Cycle* (SDLC), pemodelan berorientas objek, pemrograman berorientasi objek, bahasa pemrograman Java, metrik perangkat lunak, *majority function*, *quality rate*, dan pengujian perangkat lunak.

2.3 Rekayasa Kebutuhan

Analisis kebutuhan atau persyaratan yang dibutuhkan dalam Kakas Bantu Perhitungan Kualitas Kode Menggunakan Metrik Perangkat Lunak didapatkan permasalahan-permasalahan yang ada dari riset pustaka terhadap literatur. Tahap Rekayasa Kebutuhan pada Kakas Bantu Perhitungan Kualitas Kode Menggunakan Metrik Perangkat Lunak terdiri atas tahap elisitasi kebutuhan, tahap analisis kebutuhan, tahap deskripsi dan spesifikasi kebutuhan, serta tahap pemodelan *use case*.

Pada tahap elisitasi kebutuhan dilakukan studi terhadap literatur dan kajian pustaka untuk memperoleh kebutuhan yang dibutuhkan dalam Pembangunan Kakas Bantu Perhitungan Kualitas Kode Menggunakan Metrik Perangkat

Lunak. Pada tahap analisis kebutuhan dilakukan pendefinisian terhadap deskripsi umum sistem serta aktor dan karakteristiknya. Pada tahap spesifikasi kebutuhan dilakukan spesifikasi terhadap kebutuhan fungsional dan kebutuhan non fungsional. Pada tahap pemodelan use case dilakukan pemodelan *use case diagram* berdasarkan kebutuhan fungsional dan pendefinisian *use case scenario*. *Use case diagram* pada kakas bantu Kakas Bantu Perhitungan Kualitas Kode Menggunakan Metrik Perangkat Lunak terdapat dalam Gambar 3.



Gambar 3. *Use Case Diagram*

2.3 Perancangan

Setelah semua kebutuhan untuk Kakas Bantu Perhitungan Kualitas Kode Menggunakan Metrik Perangkat Lunak didapatkan, dilakukanlah proses perancangan dengan berorientasi objek yang terdiri dari beberapa tahapan yaitu perancangan arsitektur, perancangan algoritme, dan perancangan antarmuka. Perancangan arsitektur dilakukan dengan menggunakan pendekatan berorientasi objek atau *object oriented modelling* dan menghasilkan *sequence diagram* sesuai dengan *use case* yang telah didefinisikan, serta *class diagram* sesuai dengan objek yang terdapat dalam *sequence diagram*. Perancangan algoritme dilakukan terhadap 5 sampel algoritme yaitu algoritme Pilih Berkas Proyek, algoritme Kalkulasi, algoritme Parsing, algoritme Hitung Metriks, dan algoritme Hitung Kualitas Kode. Perancangan antarmuka dilakukan terhadap 3 rancangan antarmuka yaitu antarmuka Utama, antarmuka Pilih Berkas, dan antarmuka Proses.

2.4 Implementasi

Proses pembangunan dengan menerapkan rancangan sistem dilakukan pada tahap Implementasi. Implementasi pada Pembangunan Kakas Bantu Perhitungan Kualitas Kode Menggunakan Metrik Perangkat Lunak ini meliputi pembuatan *user interface* sebagai penghubung antara pengguna dan sistem sesuai rancangan antarmuka, serta pembangunan sistem berdasarkan perancangan arsitektur dan perancangan algoritme yang telah ada.

2.5 Pengujian

Tahap pengujian pada Kakas Bantu Perhitungan Kualitas Kode (*Quality Rate*) Menggunakan Metrik Perangkat Lunak memiliki manfaat untuk menemukan kesalahan atau *defect* pada kakas bantu, serta untuk mengetahui kesesuaian antara kebutuhan yang diimplementasikan dan kebutuhan yang diharapkan sudah sesuai. Metode pengujian yang akan dilakukan adalah dengan menggunakan metode *White-Box Testing* dan *Black-Box Testing*. Metode *White-Box Testing* terdiri atas pengujian unit dan pengujian integrasi, sedangkan metode *Black-Box Testing* terdiri atas pengujian validasi dan pengujian performa.

2.6 Penarikan Kesimpulan dan Saran

Penarikan kesimpulan akan dilakukan setelah seluruh proses perancangan dan implementasi Kakas Bantu Perhitungan Kualitas Kode (*Quality Rate*) Menggunakan Metrik Perangkat Lunak telah dilaksanakan secara keseluruhan. Kesimpulan-kesimpulan yang ada diperoleh dari analisis terhadap hasil pembangunan Kakas Bantu Perhitungan Kualitas Kode (*Quality Rate*) Menggunakan Metrik Perangkat Lunak. Kemudian tahapan terakhir dari penulisan skripsi Kakas Bantu Perhitungan Kualitas Kode (*Quality Rate*) Menggunakan Metrik Perangkat Lunak adalah saran berupa penyempurnaan penulisan dan pertimbangan keberlanjutan sistem selanjutnya.

3. PENGUJIAN DAN ANALISIS

Pengujian terhadap artefak skripsi Perhitungan Kualitas Kode (*Quality Rate*) Menggunakan Metrik Perangkat Lunak dilakukan untuk mengetahui bahwa sistem yang dikembangkan sudah sesuai dengan analisis kebutuhan dan perancangan yang telah

dilakukan sebelumnya. Penelitian ini menggunakan metode pengujian *white-box* berupa pengujian unit dan pengujian integrasi serta pengujian *black-box* berupa pengujian validasi dan performa.

Pengujian unit dilakukan pada 5 (lima) *method* yaitu chooseDirectory, calculate, parse, extractMetrics, dan computeMetrics. Pengujian unit dilakukan dengan menggunakan pendekatan *basis path testing*. Pada pengujian unit didapatkan 12 kasus uji dan persentase hasil valid sebesar 100%. Pengujian integrasi dilakukan pada *method* calculate, computeMetrics, dan extractMetrics. Pengujian integrasi dilakukan dengan pendekatan *big-bang*. Pengujian integrasi menghasilkan 8 kasus uji dan persentase hasil valid sebesar 100%.

Pengujian validasi dilakukan terhadap kebutuhan fungsional yang terdapat dalam *use case*. Pengujian validasi dilakukan menggunakan pendekatan *scenario-based testing*. Fungsi yang diuji adalah Pilih Berkas Proyek, Kalkulasi, dan Tampilkan Hasil Perhitungan. Dari pengujian validasi didapatkan persentase hasil valid sebesar 100%. Sedangkan pengujian performa dilakukan terhadap 5 (lima) data uji berupa berkas proyek Java *open source* yaitu DualSub, SoundSea, JAdventure, Aurous, dan Jasome. Dari pengujian performa didapatkan persentase hasil valid sebesar 100%.

Hasil valid pada pengujian unit, pengujian integrasi, dan pengujian validasi didapatkan apabila hasil yang diharapkan sama dengan hasil yang didapatkan. Sedangkan hasil valid pada pengujian performa didapatkan jika *run-time* perhitungan yang dilakukan oleh Kakas Bantu Perhitungan Kualitas Kode Menggunakan Metrik Perangkat Lunak kurang dari 1 menit (60 detik).

4. KESIMPULAN DAN SARAN

Kesimpulan yang dapat diperoleh dari penelitian Pembangunan Kakas Bantu Perhitungan Kualitas Kode (*Quality Rate*) Menggunakan Metrik Perangkat Lunak adalah:

1. Tahap rekayasa kebutuhan menghasilkan 3 kebutuhan fungsional yaitu Pilih Daftar Proyek yang berfungsi untuk memilih proyek Java yang akan menjadi masukan, Kalkulasi yang berfungsi untuk melakukan perhitungan untuk mendapatkan pengukuran metrik dan *quality rate*, serta Tampilkan Hasil Kalkulasi yang berfungsi untuk menampilkan hasil pengukuran metrik dan *quality rate*. Tahap ini juga menghasilkan 1 kebutuhan fungsional yaitu Performa dimana sistem harus dapat melakukan perhitungan kualitas kode kurang dari 1 menit.
2. Tahap perancangan dilakukan dengan menggunakan teori perancangan berorientasi objek (*Object-Oriented Design*) dimana perancangan menghasilkan *class diagram* dan *sequence diagram*. Terdapat 1 *class diagram* dan 3 *sequence diagram* yang sesuai dengan masing-masing kebutuhan fungsional. Selain itu perancangan juga menghasilkan perancangan algoritme dan perancangan antarmuka. Selain itu perancangan juga menghasilkan perancangan algoritme dan perancangan antarmuka. Pada perancangan algoritme terdapat 5 algoritme yaitu algoritme Pilih Berkas Proyek, algoritme Kalkulasi, algoritme Parsing, algoritme Hitung Metriks, dan algoritme Hitung Kualitas Kode. Pada perancangan antarmuka terdapat 3 antarmuka yaitu antarmuka Utama, antarmuka Pilih Berkas, dan antarmuka Proses.
3. Tahap implementasi dilaksanakan sesuai dengan rancangan arsitektur, rancangan algoritme, dan rancangan antarmuka yang telah dihasilkan sebelumnya. Kakas bantu ini diimplementasikan menggunakan bahasa pemrograman Java dan akan dijalankan pada desktop. *Library* yang dibutuhkan dalam tahap implementasi antara lain JavaFX dan ASTParser. JavaFX digunakan dalam implementasi antarmuka dan ASTParser digunakan dalam proses *parsing*.
4. Tahap pengujian dilakukan dalam 2 (dua) pendekatan yaitu pengujian white-box dan pengujian black-box. Pendekatan tersebut terdiri dari 4 metode yaitu pengujian unit, pengujian integrasi, pengujian validasi, dan pengujian performa. Pengujian unit dan pengujian integrasi tergolong sebagai pengujian *white-box*, sedangkan pengujian validasi dan pengujian performa tergolong sebagai pengujian *black-box*. Pengujian unit melibatkan 5 sampel algoritme yang terdapat pada tahap perancangan, sedangkan pengujian integrasi melibatkan 3 kelas terintegrasi pada sistem. Kedua tahap tersebut menggunakan metode *basis path*

testing. Pengujian validasi menggunakan *use case scenario* yang diperoleh dari tahap rekayasa kebutuhan sebagai acuan, sedangkan pengujian performa dilakukan terhadap data dari 5 proyek Java *open source*. Dari pengujian yang telah dilakukan, didapatkan hasil 100% valid pada pengujian unit, pengujian integrasi, pengujian validasi, dan pengujian performa.

Saran mengenai keberlanjutan penelitian Pembangunan Kakas Bantu Perhitungan Kualitas Kode (*Quality Rate*) Menggunakan Metrik Perangkat Lunak adalah diharapkan kakas bantu dapat melakukan perhitungan kualitas kode terhadap bahasa pemrograman dan metrik perangkat lunak lainnya.

5. DAFTAR PUSTAKA

Akbar, M.A., Sang, J., Khan, A.A., Fazal-E-Amin, N., Shafiq, M., Hussain, S., Hu, H., Elahi, M., dan Xiang, H. 2017. Improving the Quality of Software Development Process by Introducing a New Methodology-AZ-Model. *IEEE Access* [e-journal] 6. Tersedia melalui: IEEE Xplore <http://ieeexplore.ieee.org> [Diakses 12 Desember 2018]

Beranic, T., Podgorelec, V., dan Hericko, M. 2018. Towards a Reliable Identification of Deficient Code with a Combination of Software Metrics. *Applied Science*, [online] Tersedia melalui: <http://mdpi.com> [Diakses 12 Desember 2018]

Bouwers, E., Visser, J., dan van Deursen, A. 2012. Getting What You Measure. *Communications of The ACM*, [online] Tersedia di: <http://cacm.acm.org> [Diakses 9 Januari 2019]

Kan, S.H. 2002. *Metrics and Models in Software Quality Engineering*, 2nd ed. Boston: Addison-Wesley Longman Publishing Co., Inc.

Sharma, T., dan Spinellis, D. 2018. A survey on software smells. *Journal of Systems and Software* [e-journal] 138. Tersedia melalui: ScienceDirect <http://sciencedirect.com> [Diakses 12 Desember 2018]