STUDI LITERATUR PENGUJIAN PERANGKAT LUNAK

Dearisma Arfinda[1], Selo[2], Lukito[3]
[1,2,3]Program Studi Teknik Informatika, Fakultas Teknik, Universitas Gadjah Mada Yogyakarta
e-mail: [1]dearisma.arfinda.m@mail.ugm.ac.id, [2]selo@ugm.ac.id, [3]lukito@ugm.ac.id

ABSTRAK

Software testing atau pengujian perangkat lunak dirasa sangat penting,karena sudah banyak pihak yang mengesampingkan kegiatan ini. Ditambah lagi dengan besarnya jumlah pengguna yang mengakses aplikasi dan menggunakan aplikasi tersebut untuk memudahkan pekerjaan mereka sehari-hari. Proses pengujian ini juga bertujuan untuk memastikan kualitas dan keandalan aplikasi tersebut. Dari masalah tersebut, maka dibutuhkan metode pengujian yang tepat untuk menguji aplikasi. Ada beberapa metode untuk melakukan pengujian diantaranya seperti *metamorphic testing, random testing, adaptive random testing, state based testing, random sequence based testing.* Masing-masing metode ini menjelaskan pengertian dan kegunaannya.

Hasil dari penelitian didapat bahwa metode *metamorphic testing* dianggap paling valid dalam menemukan kesalahan untuk menguji API dan bahasa yang paling banyak digunakan untuk pengujian API adalah JSON dan JAVA, tetapi jika kasus uji berupa *object oriented programming* metode *adaptive random sequence* lebih disarankan karena metode ini lebih menekankan pada pengelompokan agar hasil yang didapat lebih detail dan akurat.

Kata Kunci*: metode pengujian, bahasa pengujian, API, contoh kasus uji*

1. PENDAHULUAN

Perkembangan teknologi pada era globalisasi ini telah memberikan dampak yang pesat pada kemajuan di berbagai aspek sosial. Tuntutan kebutuhan mengenai informasi,yang terkait dengan era globalisasi menjadi meningkat pesat dan bervariasi. Teknologi yang tadinya memanfaatkan suatu arsitektur yang besar sekarang dengan berkembang dan majunya teknologi, sistem tersebut bergerak ke dalam bentuk microservice. Microservice adalah suatu kumpulan proses independen yang saling berkomunikasi antara satu dengan yang lain bertujuan untuk membentuk aplikasi kompleks terhadap bahasa API apapun[1]. API atau sering disebut dengan *Application Programming Interface* adalah suatu wadah yang menjembatani antara pengguna dengan sistem.

Dilain sisi, banyak aplikasi atau *software* yang dibuat tetapi berhenti ditengah jalan, karena adanya kendala yang ditemukan. Sehingga menimbulkan banyak kerugian dari kendala-kendala yang muncul. Untuk mengantisipasi dampak kerugian kedepan yang ditimbulkan, maka dilakukan suatu proses yaitu pengujian. Bentuk pengujian yang digunakan dapat seperti *white box testing* atau *glass box testing*, *black box testing (behavioral testing)* dan *grey box testing*[2]. *White box testing* dilakukan pengujian berdasarkan pengecekan secara detail pada perancangan program, struktur kontrol desain program secara prosedural untuk membagi pengujian ke dalam beberapa kasus pengujian. Jika *black box testing* dilakukan pengujian dengan mengamati hasil eksekusi melalui data uji dan memeriksa fungsional dari aplikasi yang diuji.

Namun, karena di dunia industri, aplikasi yang dibuat rata-rata berkapasitas besar, maka diperlukan metode pengujian yang sederhana namun mempunyai kompleksitas yang tinggi. Sehingga diharapkan, dapat menemukan kesalahan sekecil mungkin. Nantinya, hasil dari pengujian tersebut dapat meminimasi biaya yang dikeluarkan dan siap dipasarkan.

Beberapa metode pengujian yang sering digunakan adalah *metamorphic testing*, *random testing*, *adaptive random testing.* Metode *random* testing sering digunakan dalam industri, karena kesederhanaanya. Secara umum, pendekatan pengujian yang lain memerlukan teknik pengujian yang professional karena lebih kompleks. Jika pada *metamorphic testing* user dapat memasukkan variable yang diinginkan, sehingga variable tersebut tidak random sedangkan jika pada *adaptive random testing* masukan variable random, tetapi keluaran yang sudah muncul tidak dimunculkan kembali, sehingga tidak ada data yang sama atau redundan. Pengujian *metamorphic* memberikan pandangan,jika kebenaran tidak ditentukan dengan memeriksa output dari suatu hasil tetapi dengan menerapkan transformasi input dan mengamati output yang dihasilkan sehingga valid atau tidak[3].

Pengujian dengan metode diatas sudah banyak dilakukan pada aplikasi, namun pengujian pada aplikasi berorientasi objek masih jarang dilakukan. Karena ada beberapa karakteristik khusus dari bahasa berorientasi objek misalnya seperti enkapsulasi, polimorfisme, dll. Pendekatan pengujian *object oriented software* menghasilkan sangat besar jumlah kasus uji[4].

Sedangkan khusus pada web, pengujian berarti suatu teknologi web yang menerima pesan dari pengguna dan memberikan respon yang sesuai melalui protokol atau aturan yang ada. Pertumbuhan ini juga diiringi dengan berkembangnya teknologi software development. Dengan berkembangnya teknologi yang lebih canggih ini diharapkan dapat memberikan panduan untuk menyukseskan proyek pengembangan sistem tahap demi tahap.

Untuk dapat menyelesaikan kegiatan proyek pengembangan sistem secara tahap demi tahap, ada beberapa prosedur yang dilalui dengan bergantungnya *webservice*. *Webservice* ini nantinya berfungsi untuk mentransformasi sebuah bisnis logik dan objek yang terpisah dalam satu ruang lingkup,sehingga tingkat keamanan data dapat ditangani dengan baik. Selain itu *webservice* juga memberikan kemudahan dalam proses *deployment*, karena tidak memerlukan registrasi khusus ke sistem operasi[5,6,7].

Pengujian pada *webservice* ini membawa suatu hal tantangan yang baru bagi SOA(*Service Oriented Architecture*) karena sifatnya yang dinamis dan kurangnya solusi pengujian[8].

Namun,dengan adanya karakteristik web API yang terbuka,kolaboratif,dan dinamis meningkatkan ancaman baru pada kualitas sistem[9]. Disebabkan API yang terbuka untuk akses terbuka oleh sejumlah besar pengguna di internet,kesalahan dalam API yang bersifat publik dapat tersebar luas. Hal ini menyebabkan kegagalan perangkat lunak dalam skala besar. Oleh karena itu pengujian API memegang peranan penting untuk memastikan kualitas dan keandalan API.

2. TINJAUAN PUSTAKA

Dalam melakukan menyusun studi literatur ini, penulis menggunakan suatu metode. Metode tersebut adalah dengan mengumpulkan beberapa paper dengan topik yang sejenis. Hal ini dilakukan, agar pembaca dapat lebih mudah dan memahami maksud dari penulis. Kemudian dari paper yang sudah didapat disaring dengan cara *inhale* dan *exhale*. Paper yang membahas topik yang sama akan masuk ke kriteria *inhale*, sedangkan paper yang membahas topik berbeda akan masuk ke kriteria *exhale*.

Pencarian paper terkait dilakukan sesuai dengan topik yang diangkat pada paper ini. Paper ini nantinya akan memudahkan melakukan pengujian di aplikasi web (memanfaatkan API) dan aplikasi desktop (program OOP).

Topik yang akan dibahas disini adalah mengenai software testing, lebih detailnya mengenai pentingnya melakukan software testing, metode software testing, bahasa software testing, dan studi kasus yang dilakukan selama tahun 2015-2019.

3. METODE PENELITIAN

Melakukan proses pengujian, merupakan hal yang penting dari suatu aplikasi atau produk sebelum muncul ke pasaran. Proses pengujian ini bertujuan untuk memastikan kualitas dan keandalan aplikasi tersebut

Aplikasi berbasis *object oriented* semakin berkembang dan banyak digunakan. Namun, pengujian pada *object oriented program* menghadapi tantangan untuk menerapkan pengujian tradisional ke *object oriented program*. Pada *object oriented program* terdapat beberapa karakteristik khusus dari segi bahasa seperti enkapsulasi, polimorfisme,dan pewarisan [4].

Sedangkan karakteristik web API adalah terbuka,kolaboratif,dan dinamis meningkatkan ancaman baru pada kualitas sistem[9]. Hal itu disebabkan API yang terbuka untuk akses terbuka oleh sejumlah besar pengguna di internet,kesalahan dalam API yang bersifat publik dapat tersebar luas. Hal ini juga menyebabkan kegagalan perangkat lunak dalam skala besar. Pengujian API dengan demikian menjadi perlu untuk memastikan kualitas dan kendalan API [9].

Maka, dengan masalah diatas dibutuhkan pengujian pada aplikasi, agar nantinya aplikasi dapat berjalan dengan baik. Ada beberapa metode pengujian pada paper ini, bahasa pengujian yang sering digunakan, dan contoh studi kasus. Sehingga, nanti penguji dapat mengetahui metode-metode dan bahasa yang cocok untuk melakukan pengujian.

4. HASIL DAN PEMBAHASAN

4.1 Metode Pengujian

4.1.1 Metamorphic Testing

Metamamorphic testing diusulkan untuk mengatasai masalah-masalah bug yang terjadi pada oracle [10,11], karena saat menguji program tanpa uji oracle adalah hal yang mustahil untuk mengetahui apakah test case menyebabkan kegagalan atau tidak[12]. MT menggunakan *metamorphic relation* (MR) untuk menentukan kasus uji atau *test case*, yang nantinya akan menghasilkan test-test dari *case* sumber tes input itu sendiri [12,13,14,15]. Kasus uji ini berguna untuk memudahkan penguji dalam melakukan pengujian. Contohnya adalah pengujian pada masukan input di spotify apakah outputnya sesuai dengan hasil input atau tidak[16].

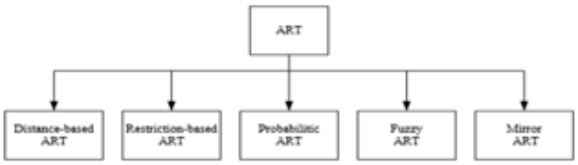
4.1.2 Random Testing

Random Testing adalah metode pengujian perangkat lunak yang banyak digunakan karena kemudahan dan kepraktisannya. Metode ini sangat sederhana dan mudah untuk di otomatissi [15,17].

Random Testing dilakukan dengan cara memasukkan variabel masukan (test case) dan akan di seleksi secara acak. Dari inputan tersebut, akan didapatkan hasil yang dimana akan ada proses pengacakan dulu. Metode ini merupakan metode paling mudah dan simple, untuk itu pengujian ini tidak cocok dengan aplikasi berskala besar.

4.1.3 Adaptive Random Testing

Adaptive Random Testing memiliki fokus yang mempengaruhi jarak pada kinerja pengujian acak dan efisiensi pendeteksian kesalahan [17,18]. Cara pengujian adaptive random testing ini dengan memberikan input uji yang acak dan menyebar menggunakan pengukuran jarak antara input yang berurutan [17,19], sehingga hasil input yang salah dapat dikelompokkan. Kesalahan mendeteksi kemampuan adaptif pengujian acak yang terjadi dipengaruhi oleh kesalahan penghitungan jarak dan kesalahan pemodelan pola kegagalan blok [17]. Metode adaptive random testing sudah mengalami perkembangan, yang dapat dilihat pada gambar 1.



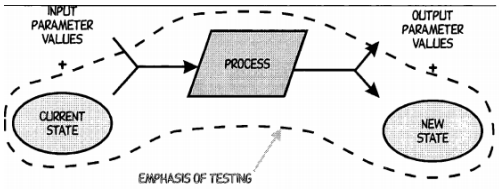
Gambar 1. Perkembangan Adaptive Random Testing *(ART)*

4.1.4 Mirror Adaptive Random Testing

Mirror Adaptive Random Testing merupakan pembaharuan metode dari *adaptive rando testing.* Sekilas cara kerja metode ini sama, yaitu dengan menggunakan kasus uji (*test case)*. Metode ini pun tetap menggunakan algoritma *adaptive random testing* sebagai acuannya. Perbedaanya adalah pada metode ini kasus uji dibagi menjadi beberapa subdomain, dan salah satu subdomain dijadikan subdomain utama dan dapat dikerjakan dengan menggunakan metode *adaptive random testing.* Jika hasil dari kasus uji subdomain utama tersebut tidak ditemukan eror, maka kasus uji diteruskan dengan cara mencerminkannya pada subdomain yang lain [20].

4.1.5 State Based Testing

State-based testing adalah suatu metode baru untuk menguji program berorientasi objek. terbatas. Setiap kelas yang diuji diibaratkan sebagai pemetaan dari status awal ke status penyelesaian tergantung pada parameter yang diteruskan [21]. Jadi metode ini melakukan pengujian dengan memvalidasi interaksi antara operasi dan keadaan suatu objek. Tetapi *state based testing* berkaitan dengan nilai aktual yang disimpan oleh atribut objek. Hal ini tidak secara langsung berkaitan dengan validasi kombinasi nilai parameter tertentu, meskipun nilai ini digunakan untuk memvalidasi transisi status objek, yang dapat dilihat pada gambar 2.



Gambar 2. State Based Testing

Metode *state-based testing* ini memiliki beberapa kelebihan dibandingkan dengan pemodelan konseptual, khususnya kemudahan manipulasi nilai-nilai fisik dan kemandirian setiap operasi dari operasi lain yang disediakan oleh suatu objek.

4.1.6 Random Sequence-based Testing

Adaptive Random Testing (ART) merupakan suatu metode untuk pengujian dengan menyebarkan input tes secara merata ke seluruh domain input menggunakan metrik kesamaan atau tidak kesamaan . Hasil dari ART tidak hanya untuk menghasilkan urutan sendiri dari kasus uji, tetapi juga dapat memberikan rangkaian uji untuk meningkatkan peluangnya mendeteksi kegagalan sebelumnya, dengan urutan terurut yang disebut dengan *Adaptive Random Sequence* (ARS) [4].

ARS bekerja dengan menyebarkan seluruh domain input dan lebih efektif untuk menemukan kesalahan atau kegagalan lebih cepat. Penerapan metode ARS pada pengujian regresi, dan dapat berupa urutan alternatif ke urutan acak sederhana, yang biasanya digunakan pada pengujian regresi. Untuk itu ARS sangat efektif untuk meningkatkan kinerja pengujian regresi untuk OOS. Metode ini dapat diuji berdasarkan :

1. Clustering

Clustering adalah proses untuk mengelompokkan data ke beberapa *cluster* (kelompok) sehingga data dalam satu *cluster* memiliki tingkat kemiripan yang banyak dan data antar *cluster* memiliki tingkat kemiripan yang sedikit [22].

Clustering dilakukan dengan proses partisi satu set objek data ke dalam himpunan bagian yang disebut *cluster*. Objek dalam *cluster* ini memiliki kemiripan karakteristik antara satu dengan lainnya dan berbeda dengan *cluster* lain. Partisi dilakukan menggunakan algoritma *clustering.* Sehingga *clustering* sangat berguna dan bisa menemukan kelompok yang tidak dikenal dalam data. *Clustering* banyak digunakan dalam berbagai aplikasi seperti pada aplikasi *business intelligence*, pengenalan pola citra, *web search*, bidang ilmu biologi, dan

bidang keamanan (*security).* Ada beberapa metode algoritma pada *clustering*, misalnya adalah algoritma K-means dan K-medoids.

2. K-means

K-means adalah suatu algoritma bagian dari *clustering* dengan menggunakan metode analisis kelompok yang mengarah pada pemartisian N objek pengamatan ke dalam K kelompok, dimana setiap objek pengamatan dimiliki oleh sebuah kelompok dengan nilai *mean* (rata-rata) terdekat. Dimana keduanya akan menemukan pusat dari kelompok dalam data sebanyak iterasi perbaikan yang dilakukan [23].

3. K-medoids

K-medoids adalah suatu algoritma atau sering disebut dengan PAM (*Partitioning Around Medoids)* menggunakan metode partisi *clustering* untuk mengelompokkan sekumpulan n objek menjadi sejumlah k *cluster*. Objek yang menjadi wakil dari sebuah *cluster* disebut dengan *medoid. Cluster* dibangun dengan menghitung kedekatan yang dimiliki antara *medoid* dengan objek *non-medoid.* Metode ini dapat mengerjakan inputan berupa numerik [4].

4.2 BAHASA PENGUJIAN

Bahasa yang dimaksud adalah bahasa pemograman untuk software testing itu sendiri. Ada beberapa jenis bahasa pemograman yang biasa digunakan untuk membuat program software testing diantaranya adalah :

4.2.1 Bahasa C#

Bahasa C# adalah bahasa yang dikembangkan oleh Microsoft. Bahasa ini banyak digunakan, karena kemudahan untuk memahami dan prakteknya.Bahasa ini adalah bahasa pemograman yang berfokus pada orientasi objek. Bahasa pemograman ini membantu untuk proses pengujian yang membutuhkan objek, karena lebih memudahkan dalam prosesnya. Hasil yang didapatkan juga lebih mudah, karena memiliki fokus yang sama yaitu berorientasi objek [4].

4.2.2 Javascript

Javascript menurunkan bahasa dengan format data JSON, keunggulan JSON memiliki banyak format yang dapat diakses dengan bahasa pemoraman Python,Ruby,PHP, dan Java. Selain itu, JSON memiliki library untuk membantu pemograman khususnya untuk pengujian menggunakan metode yang memanfaatkan API suatu website dengan menyediakan fungsi yang lebih fleksibel dan umumnya memilih eksekusi interpretative untuk bahasa script yang dapat menyederhanakan pemrosesan respon [7,16].

4.2.3 JAVA

JAVA adalah Bahasa pemograman yang paling banyak digunakan. Dengan Bahasa pemograman JAVA, dapat memberikan hasil berupa nilai-nilai yan dapat membedakan antara yang asli dan mutase [24]. Untuk membantu memudahkan mendapatkan hasil pada proses pengujian dibutuhkan test case, yang kemudian akan di proses dengan Bahasa pemograman ini. Sehingga hasil dari porses pengujian lebih akurat dan memberikan kemudahan selama proses pengujian berlangsung. Kebanyakan bahasa ini dipadukan dengan metode pengujian *metamorphic testing.*

4.3 CONTOH KASUS UJI

4.3.1 TASSA : Testing Framework for Webservice Orchestrations

Pada paper ini mengangkat studi kasus bisnis sampel proses untuk belanja online (*shopping card).* Pengujian ini berfokus pada layanan arsitektur perangkat lunak (TASSA). Hasil pengujian TASSA terhadap proses bisnis tersebut dianggap layak dengan memberikan manfaat seperti :

a. Kemampuan untuk menguji proses bisnis bahkan jika web mitra layanan tidak tersedia atau sedang dikembangkan menyediakan fungsionalitas untuk isolasi ketergantungan.
b. Kemampuan untuk melakukan pengujian ketahanan melalui disediakan fungsionalitas untuk injeksi kesalahan.
c. Kemampuan untuk mengurangi upaya pengujian melalui disediakan fungsionalitas untuk pembuatan template uji dari yang sudah ada.

Tujuan penelitian ini tidak hanya memverifikasi kualitas dari proses bisnis, tetapi untuk menemukannya pola untuk memprediksi kemungkinan kegagalan [8].

4.3.2 Metamorphic Testing of Restful Web API

Pada paper ini mengangkat studi kasus pengujian pada spotify dan youtube dengan menggunakan metode *metamorphic testing.* Ada 6 pola pengujian yang digunakan yaitu *equivalence,equality, subset,disjoint,complete,* dan *difference.* Setiap pola didefinisikan dalam hal hubungan yang diatur antara API tanggapan dan dapat digunakan menjadi satu atau lebih konkrit hubungan metamorfik di masing-masing API web yang sedang diuji..Dari pengujian tersebut didapatkan hasil 2 pelanggaran dari 20 pada spotify dan 19 pelanggaran dari 40 pada youtube [16].

4.3.3 Metamorphic Testing of Navigation Software : A Pilot Study with Google Maps

Pada paper ini mengangkat studi kasus pada sistem navigasi yaitu google maps dengan menggunakan metode *metamorphic testing*. Saat melakukan tes pada situs web GUI di maps.google.com, didapat hasil bahwa google maps GUI dan API tidak sama. API lebih rentan terhadap masukan 'besar' yang melibatkan banyak *node waypoint*. Kesalahan yang ditemukan pada pengujian ini sudah dilaporkan ke pihak google, dan pihak google sedang memproses lebih lanjut [13].

4.3.4 Test Case Priorization for Object-Oriented Software : An Adaptive Random Sequence Approach Based on Clustering

Pada paper ini mengangkat studi kasus pada *object-oriented software* dengan menggunakan metode *adaptive random sequence.* Tujuannya untuk meningkatkan *test case prioritization*(TCP), karena TCP dapat meningkatkan tingkat deteksi kesalahan dengan mengoptimalkan urutan uji kasus uji, sehingga kasus uji yang lebih penting dijalankan diawal. Penelitian ini menggunakan 3 metode pengelompokan yaitu *clustering means,clusteringmedoid,* dan *clustering* menggunakan algoritma K-means dan K-medoids. Hasil yang didapat bahwa ketiga metode tersebut lebih efisien terutama untuk pengujian *object-oriented* software dalam sekala besar. Direkomendasikan 15% untuk program besar dan 10% untuk program kecil [4].

4.3.5 Pemanfaatan Mirror Adaptive Random Testing untuk Uji Coba Software

Pada paper ini melakukan pengujian dengan menggunakan pendekatan *black box testing* pada aplikasi CV. Mitra Informatika Solusindo. Dalam penelitian ini dilakukan perbandingan dua metode, yaitu *adaptive random testing* dan *mirror adaptive random testing.* Pengujian memanfaatkan 3 kasus dengan domain input yang berbeda yaitu 1 domain input, 2 domain input, dan 4 domain input.Hasil yang didapat adalah bahwa metode *adaptive random testing* dan *mirror adaptive random testing* dapat dilakukan dengan menggunakan kasus uji yang sama, dan dapat meningkatkan kualitas aplikasi tersebut untuk menemukan F-measure diatas 5% yang diterapkan dalam 10 kali pengujan pada setiap iterasi. Tetapi F-measure tidak selalu sama pada setiap pengujian, *adaptive random testing* karena bergantung pada hasil pengacakan kasus uji [20].

4.3.6 Metamorphic Testing for Adobe Analytics Data Collection JavaScript Library

Pada paper ini berdiskusi tentang Adobe Analytics Data Collection Javascript Library. Berbagai kasus uji perlu digunakan untuk menghasilkan banyak kombinasi kasus uji pada pengujian, tetapi metode konvensional sudah dilakukan dan didapat hasil yang tidak efektif dan efisien.Untuk itu diusulkan menggunakan metode *metamorphic testing*. Hasil dari penelitian ini adalah adanya dua bug dan satu masalah kompatibilitas yang melibatkan browser Microsoft IE 11. Selain itu dua MR yang digunakan sangat sederhana, sehingga mungkin tidak terlalu besar pengaruhnya dalam mencari kesalahan [25].

4. TANTANGAN PENELITIAN

Dari pembahasan diatas, metode untuk software testing yang paling banyak digunakan adalah *metamorphic testing.* Metode ini digunakan, karena banyak yang memanfaatkan API untuk pengujian. Hal itu dilakukan karena dirasa lebih mudah dan praktis. Tetapi, untuk metode yang paling akurat dari semua metode yang sudah dibahas adalah metode *random sequence based testing.*

Setiap aplikasi yang berjalan di desktop dan web memiliki metode *testing* yang berbeda. Jika aplikasi tersebut berbasis web, maka akan lebih mudah melakukan testing, karena dapat memanfaatkan API dari web tersebut, dan biasanya bahasa pemograman yang digunakan adalah JSON, dan Java. Sedangkan untuk aplikasi berbasis desktop, memiliki tingkat *testing* yang kompleks, dan bahasa pemograman yang digunakan adalah C#. Seiring, berjalannya waktu, pengujian berbasis desktop jarang dilakukan, penelitian lebih sering dilakukan dengan memanfaatkan API.

Selain itu, perkembangan *software testing* kedepan adalah pada bagian *microservice*, karena hal ini memudahkan struktur system aplikasi berjalan lebih mudah dan tidak kompleks. Hal ini menantang *software testing* kedepan, untuk menguji *microservice* dengan akurat dan mudah. Jika kedepannya, pada *software testing* dapat melakukan pengujian dengan metode yang sama antara aplikasi berbasis desktop dan web, karena sekarang ini banyak aplikasi yang dibuat berbasis desktop dan web. Sejauh ini, belum ada yang membahas bagaimana melakukan pengujian dengan satu metode yang sama untuk menguji aplikasi yang sama di basis yang berbeda dengan satu waktu yang sama.

5. KESIMPULAN

Software testing dirasa sangat penting,karena sudah banyak pihak yang mengesampingkan kegiatan ini. Ditambah lagi dengan besarnya jumlah pengguna yang mengakses aplikasi dan menggunakan suatu aplikasi untuk memudahkan pekerjaan mereka sehari-hari.

Ada beberapa metode pengujian yang dijelaskan seperti *metamorphic testing, random testing, adaptive random testing, state based testing, random sequence based testing.* Dari beberapa metode tersebut, terdapat kelebihan dan kelemahan masing-masing. Tetapi metode *random sequence based testing* adalah metode yang paling akurat, karena dalam melakukan pengujian dengan detail. Tetapi metode ini dikhususkan untuk *object oriented programming*, jadi jika ingin melakukan pengujian terhadap API, metode *metamorphic testing* mungkin metode yang tepat untuk melakukan pengujian.

DAFTAR PUSTAKA

[1] V. Heorhiadi, S. Rajagopalan, H. Jamjoom, M. K. Reiter, and V. Sekar, "Gremlin: Systematic Resilience Testing of Microservices," *Proc. - Int. Conf. Distrib. Comput. Syst.*, vol. 2016-Augus, pp. 57–66, 2016.

[2] K. Sneha and G. M. Malle, "Research on software testing techniques and software automation testing tools," *2017 Int. Conf. Energy, Commun. Data Anal. Soft Comput. ICECDS 2017*, pp. 77–81, 2018.

[3] S. Segura, G. Fraser, A. Sanchez, and A. Ruiz-Cortés, "A Survey on Metamorphic Testing," *IEEE Trans. Softw. Eng.*, vol. 42, no. 9, pp. 805–824, 2016.

[4] J. Chen *et al.*, "Test case prioritization for object-oriented software: An adaptive random sequence approach based on clustering," *J. Syst. Softw.*, vol. 135, no. 2018, pp. 107–125, 2018.

[5] D. Petrova-Antonova, K. Kuncheva, and S. Ilieva, "Automatic Generation of Test Data for XML Schema-based Testing of Web Services," pp. 277–284, 2015.

[6] J. Liu and W. Chen, "Optimized Test Data Generation for RESTful Web Service," *Proc. - Asia-Pacific Softw. Eng. Conf. APSEC*, vol. 2017-Decem, pp. 683–688, 2018.

[7] W. Hu, Y. Huang, X. Liu, and C. Xu, "Study on REST API Test Model Supporting Web Service Integration," *Proc. - 3rd IEEE Int. Conf. Big Data Secur. Cloud, BigDataSecurity 2017, 3rd IEEE Int. Conf. High Perform. Smart Comput. HPSC 2017 2nd IEEE Int. Conf. Intell. Data Secur.*, pp. 133–138, 2017.

[8] D. Petrova-Antonova, S. Ilieva, and D. Manova, "TASSA: Testing Framework for Web Service Orchestrations," *Proc. - 10th Int. Work. Autom. Softw. Test, AST 2015*, pp. 8–12, 2015.

[9] J. Wang, X. Bai, L. Li, Z. Ji, and H. Ma, "A Model-Based Framework for Cloud API Testing," *Proc. - Int. Comput. Softw. Appl. Conf.*, vol. 2, pp. 60–65, 2017.

[10] M. Srinivasan, M. P. Shahri, U. Kanewala, and I. Kahanda, "Quality Assurance of Bioinformatics Software: A Case Study of Testing a Biomedical Text Processing Tool Using Metamorphic Testing," *2018 IEEE/ACM 3rd Int. Work. Metamorph. Test.*, pp. 26–33, 2018.

[11] M. Troup, A. Yang, A. H. Kamali, E. Giannoulatou, T. Y. Chen, and J. W. K. Ho, "A cloud-based framework for applying metamorphic testing to a bioinformatics pipeline," *2016 IEEE/ACM 1st Int. Work. Metamorph. Test.*, pp. 33–36, 2016.

[12] Z. W. Hui, S. Huang, T. Y. Chen, M. F. Lau, and S. Ng, "Identifying failed test cases through metamorphic testing," *Proc. - 2017 IEEE 28th Int. Symp. Softw. Reliab. Eng. Work. ISSREW 2017*, pp. 90–91, 2017.

[13] J. Brown, Z. Q. Zhou, and Y.-W. Chow, "Metamorphic Testing of Navigation Software: A Pilot Study with Google Maps," *Proc. 51st Hawaii Int. Conf. Syst. Sci.*, vol. 9, 2018.

[14] A. C. Barus, T. Y. Chen, F.-C. Kuo, H. Liu, and H. W. Schmidt, "The impact of source test case selection on the effectiveness of metamorphic testing," pp. 5–11, 2016.

[15] E. Nikravan, F. Feyzi, and S. Parsa, "Enhancing path-oriented test data generation using adaptive random testing techniques," *Conf. Proc. 2015 2nd Int. Conf. Knowledge-Based Eng. Innov. KBEI 2015*, pp. 510–513, 2016.

[16] S. Segura, J. A. Parejo, J. Troya, and A. Ruiz-Cortes, "Metamorphic testing of RESTful Web APIs," *IEEE Trans. Softw. Eng.*, vol. 44, no. 11, pp. 1083–1099, 2018.

[17] Y. Qi, Z. Wang, and Y. Yao, "Influence of the Distance Calculation Error on the Performance of Adaptive Random Testing," *Proc. - 2017 IEEE Int. Conf. Softw. Qual. Reliab. Secur. Companion, QRS-C 2017*, pp. 316–319, 2017.

[18] K. K. Sabor and S. Thiel, "Adaptive Random Testing by Static Partitioning," *Proc. - 10th Int. Work. Autom. Softw. Test, AST 2015*, pp. 28–32, 2015.

[19] J. Chen, F. C. Kuo, T. Y. Chen, D. Towey, C. Su, and R. Huang, "A similarity metric for the inputs of OO programs and its application in adaptive random testing," *IEEE Trans. Reliab.*, vol. 66, no. 2, pp. 373–402, 2017.

[20] S. Jatmika and E. Pramana, "Pemanfaatan Mirror Adaptive Random Testing Untuk Uji Coba Software," 2015.

[21] C. D. Turner, "State-based testing - a new method for testing object-oriented programs," 1994.

[22] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction of Data Mining*. Boston: Pearson Education, 2006.

[23] E. Prasetyo, *Data Mining Konsep dan Aplikasi Menggunakan Matlab*. Yogyakarta: Andi, 2012.

[24] P. Saha and U. Kanewala, "Fault detection effectiveness of source test case generation strategies for

metamorphic testing," *2018 IEEE/ACM 3rd Int. Work. Metamorph. Test.*, pp. 2–9, 2018.
[25] Z. Wang, D. Towey, Z. Q. Zhou, and T. Y. Chen, "Metamorphic testing for adobe analytics data collection javascript library," *2018 IEEE/ACM 3rd Int. Work. Metamorph. Test.*, no. 1, pp. 34–37, 2018.