p-ISSN 2477-3506
e-ISSN 2549-1938

SOFTWARE UNDER TEST DALAM PENELITIAN SOFTWARE TESTING: SEBUAH REVIEW

Joe Lian Min[1], Suprihanto[2], Ani Rahmani[3]

[1,2,3]Politeknik Negeri Bandung

[1]joelianmin@jtk.polban.ac.id, [2]sprh@jtk.polban.ac.id, [3]anirahma@jtk.polban.ac.id

Abstrak

Abstrak-- Software under Test (SUT) merupakan hal penting dalam kegiatan penelitian *software testing* (pengujian perangkat lunak). Penyiapan SUT tidaklah sederhana, diperlukan kecermatan, lengkap, dan sesuai dengan tujuan penelitian, dan akan mempengaruhi kualitas penelitian yang dilakukan. Di satu sisi saat ini terdapat sejumlah cara untuk membangun SUT, mulai membangun sendiri, hingga SUT yang secara langsung dapat dimanfaatkan oleh peneliti. Artikel ini membahas hasil identifikasi SUT dalam sejumlah penelitian, serta tren penggunan SUT yang dilakukan oleh para peneliti software testing. Hasil dari penelitian adalah berupa informasi, tentang tren, penggunaan SUT dari open source, dan SUT dari repositori. Penelitian yang dilakukan berupa *systematic literature review* (SLR), menggunakan protocol dari Kitchenham. Proses review dilakukan terhadap 86 artikel yang dipublikasi tahun 2017-2020. Artikel tersebut terpilih setelah melalui dua tahap seleksi yaitu *inclusion* dan *exclusion criteria*, serta asesmen atas kualitas dari sisi kejelasan SUT pada artikel yang diseleksi. Hasil studi menunjukkan bahwa tren pengguanaan *open source* untuk digunakan sebagai SUT sangat dominan. Beberapa peneliti menggunakan *open source* tersebut sebagai basis dikembangkannya SUT, sementara peneliti lain memanfaatan SUT dari repositori yang menyediakan SUT siap pakai. Dalam hal ini, penggunaan SUT dari *software infrastructure repository* (SIR) dan Defect4J menjadi pilihan para peneliti dalam jumlah signifikan.

Kata Kunci: *software testing, software under test (SUT), open source, repository SUT*

Abstract

Software under Test (SUT) is an essential aspect of software testing research activities. Preparation of the SUT is not simple. It requires accuracy, completeness and will affect the quality of the research conducted. Currently, there are several ways to utilize an SUT in software testing research: building an own SUT, utilization of open source to build an SUT, and SUT from the repository utilization. This article discusses the results of SUT identification in many software testing studies. The research is conducted in a systematic literature review (SLR) using the Kitchenham protocol. The review process is carried out on 86 articles published in 2017-2020. The article was selected after two selection stages: the Inclusion and Exclusion Criteria and the quality assessment. The study results show that the trend of using open source is very dominant. Some researchers use open source as the basis for developing SUT, while others use SUT from a repository that provides ready-to-use SUT. In this context, utilization of the SUT from the software infrastructure repository (SIR) and Defect4J are the most significant choice of researchers.

Keywords: *software testing, software under test (SUT), open source, SUT repository*

I. PENDAHULUAN

Software testing (*testing*) adalah salah satu tahap penting dalam pengembangan perangkat lunak. Menurut L. Luo (2000) *effort* yang diperlukan untuk melaksanakan testing adalah sekitar 40-50% dari keseluruhan *effort* pengembangan perangkat lunak (Luo, 2000). Sementara dari sisi biaya, testing menyerap biaya paling tinggi dibanding tahapan lain dalam software maintenance (Hynninen et al., 2018). Tujuan utama *testing* adalah untuk menjamin software yang dikembangkan memiliki standar kualitas tertentu sesuai spesifikasi yang didefinisikan. Aktivitas testing secara global meliputi perancangan *test-case*, mengeksekusi *software* yang diuji, dan memeriksa kesesuaian hasil *testing* dengan *requirement* (Dadkhah et al., 2020; Garousi et al., 2019)

Dalam konteks kegiatan penelitian, salah satu aspek penting dalam penelitian *software testing* adalah penyiapan *software under test* (SUT), yaitu sebuah *software* yang akan menjadi objek pengujian (*testing*).

SUT dikatakan penting dalam penelitian *software testing*, karena kualitas SUT akan turut menentukan kualitas penelitian. SUT yang baik diperlukan untuk menjamin eksperimen yang dilakukan menjadi valid. Di samping itu, studi mengenai SUT dipandang penting, karena secara umum penyiapan SUT dalam sebuah penelitian membutuhkan *effort* yang tidak sederhana. Di sisi lain, sejauh ini penelitian di bidang *software testing* terus berkembang, dan dalam realitasnya terdapat banyak *open-source* yang dapat dirancang untuk menjadi SUT.

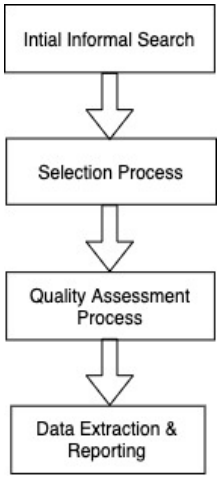
Identifikasi mengenai SUT, dilakukan hanya berupa bagian dari survey terhadap pendekatan atau teknik tertentu dalam penelitian *software testing*. Misalnya, survey terhadap *test case prioritization* oleh (Hasnain et al., 2021) dan (Samad et al., 2021). Di sisi lain, terdapat peneliti yang secara khusus mengkaji suatu repositori SUT, seperti yang dilakukan oleh Gay, G., & Just, R. (2020) yang mengkaji repository Defect4J (Gay and Just, 2020) dan (Sobreira et al., 2018).

Artikel ini membahas hasil identifikasi penggunaan SUT dalam penelitian *software testing*. Studi yang dilakukan berupa *systematic literature review* (SLR) yang diperkenalkan oleh Kitcehnmant (2013). Hasil studi diharapkan dapat memberikan manfaat bagi para peneliti *software testing*, salah satunya memanfaatkan *open source* untuk SUT, sehingga para peneliti dapat lebih fokus pada inti yang dikaji, dan penyiapan SUT tidak lagi memerlukan *effort* terlalu tinggi.

Pembahasan setelah pendahuluan, dibagi ke dalam beberapa sub bagian, meliputi: metode, hasil dan pembahasan, serta penutup.

II. METODE

Dalam artikelnya, Kinchenham (2013) telah melakukan studi berupa SLR untuk bidang *software engineering* (Kitchenham and Brereton, 2013). Hingga saat ini, artikel tersebut telah dirujuk ratusan peneliti untuk melakukan SLR di bidang *software engineering*. Gambar 1 adalah langkah-langkah global SLR dari Kinchenham.



Gambar 1. Langkah-langkah SLR

Initial Search

Pada tahap initial, dilakukan perumusan *research question* (RQ). Terdapat 3 RQ yang akan dijawab dalam aktivitas SLR ini, yaitu:

1. Bagaimana tren penggunaan SUT dalam penelitian *software testing*? (RQ1)
2. Bagaimana karakteristik *open source* untuk SUT yang digunakan para peneliti *software testing*? (RQ-2)
3. Bagaimana karakteristik *open source* SUT yang tersedia pada database repositori SUT? (RQ3).

Selection Process

Pada tahap ini, dilakukan seleksi terhadap artikel yang memenuhi beberapa kriteria. Kriteria tersebut ditentukan dengan tujuan tertentu, yang dimulai dari pemilihan database sebagai sumber artikel.

Penentuan Data Source

Sumber pustaka yang digunakan bersumber dari beberapa database yang secara umum dipandang kredibel dan berpengaruh. Pencarian dilakukan dengan mengkombinasi dua acara yaitu manual dan

otomatis melalui *tools* layanan pada database tersebut. Database yang menjadi sumber pencarian artikel yaitu:
a. IEEE Xplore
b. Science Direct
c. Springer
d. Google Scholar

Search String Criteria

Pada tahap ini dilakukan penentuan string yang menjadi keyword dalam proses pencarian artikel. Beberapa string tersebut adalah: "*empirical software testing", "software testing research", "software testing approach", "regression testing", "unit testing technique"*. Namun dalam pelaksanaannya *keyword* tersebut kurang memperoleh hasil signifikan, karena terlalu umum, sehingga digunakan *keyword* lain yang lebih spesifik, dengan menyertakan teknik atau pendekatan *software testing* pada proses pencarian, misalnya: "*search-based testing", "model-based testing", "test case selection"*, dan lain-lain.

Inclusion/Exclusion Criteria

Inclusion dan *exclusion* criteria adalah upaya untuk menyeleksi artikel yang akan dipilih (*Inclusion Criteria*) dengan yang tidak (*Exclusion Criteria*), dengan cara mendefinisikan kriteria masing-masing. Pada SLR ini inclusion dan *exclusion criteria* diperlihatkan pada Tabel 1. Selanjutnya dilakukan penilaian terhadap kualitas artikel hasil IC dan EC, dengan melihat kejelasan deskripsi mengenai SUT.

Data Extraction and Reporting

Pada aktivitas ini dilakukan penentuan artikel terpilih, untuk selanjutnya dilakukan klasifikasi dan analisis terhadap artikel-artikel tersebut.

Tabel 1. *Inclusion Criteria (IC)* dan *Exclusion Criteria* (EC)

#IC	Inclusion Criteria	#EC	Exclusion Criteria
IC1	Artikel yang dipilih berasal dari jurnal atau prosiding, yang merupakan hasil studi menggunakan teknik testing tertentu	EC1	Lecture note atau book chapter, berupa diskusi atau overview sebuah konsep
IC2	Artikel dipublikasi dari tahun 2017-2020	EC2	Artikel dipublikasi di luar rentang 2017-2020
IC3	Artikel ditulis dalam Bahasa Inggris	EC3	Artikel dalam Bahasa selain Inggris

III. HASIL DAN PEMBAHASAN

Bagian ini menjelaskan proses seleksi artikel, tren penggunaan SUT, *open source* sebagai SUT, dan repositori SUT untuk menjawab semua RQ.

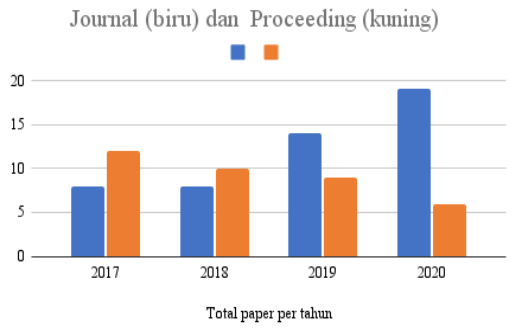
Pencarian Artikel

Hasil pencarian artikel, pada tahap awal diperoleh 1241 artikel. Jumlah tersebut kemudian diseleksi menggunakan kriteria *Inclusion* (IC) dan *Exclusion* (EC). Hasil seleksi dicantumkan pada Tabel 2. Sejumlah artikel di-*exclude*, diantaranya karena berupa lecture note atau artikel hasil *review/overview* (bukan hasil penelitian), dan dipublikasi di luar rentang 2017-2020. Dari proses seleksi tersebut diperoleh 267 artikel.

Dalam hal seleksi terhadap kualitas artikel dilakukan dengan melihat kejelasan SUT. Terdapat sejumlah artikel yang tidak secara jelas menginformasikan SUT. Seleksi tahap kedua menghasilkan 86 artikel yang menjadi dasar dari studi ini. Gambar 2 memperlihatkan sebaran artikel hasil seleksi yang dikelompokkan per tahun.

Tabel 2. Daftar Artikel Menurut Penerbitan

Data Artikel Terpilih	Journal	Proceeding
2017	8	12
2018	8	10
2019	14	9
2020	19	6
Jumlah	49	37



Gambar 2. Sebaran Artikel Terpilih

Tren Penggunaan SUT

Untuk menjawab RQ1 (Bagaimana tren

penggunaan SUT dalam penelitian software testing?), diperoleh gambaran sebagai berikut. Secara umum para peneliti melakukan uji coba menggunakan berbagai variasi SUT. Beberapa peneliti menggunakan SUT skala kecil, untuk lingkup pembelajaran di kelas, misalnya aplikasi ATM, dengan hanya beberapa fitur. Hanya sedikit peneliti yang menggunakan SUT berskala besar dari industri. Dalam mengidentifikasi jenis SUT, telah dilakukan klasifikasi terhadap SUT berdasarkan 6 kelompok. Tabel 3 memperlihatkan hasil identifikasi dan klasifikasi SUT dari 86 artikel penelitian *software testing*.

Tabel 3. Klasifikasi SUT dalam Penelitian

No	Software Under Test (SUT)	#Artikel Penelitian	Jumlah
1	Industrial System	(Shin and Lim, 2020) (Alfieri et al., 2020) (Abbas et al., 2019) (Anderson et al., 2019) (Khanna et al., 2019a) (Hajri et al., 2020) (Magalhães et al., 2020) (Azizi and Do, 2018a) (Ouriques et al., 2018) (Ulewicz et al., 2018) (Matinnejad et al., 2019) (Chen et al., 2018) (Arrieta et al., 2018) (Riskiawan and Azhari, 2017) (Matinnejad et al., 2017)	15
2	Membangun SUT sendiri	(Yu et al., 2019) (Mishra et al., 2019) (Zhang et al., 2018) (Bosmans et al., 2019) (Lei Xiao, Huaikou, Weiwei Zhuang, 2017) (Vasanthapriyan et al., 2017) (Chuaychoo and Kansomkeat, 2017) (Sinaga, 2017) (Khanna et al., 2019a)(Rahmani et al., 2020)	10
3	Membangun SUT dari open source	(Huang et al., 2020a) (H. Wang et al., 2020) (Eldrandaly et al., 2020) (Mohd-Shafie et al., 2020) (Li et al., 2020) (Rahmani et al., 2021) (Butool et al., 2019) (Hettiarachchi and Do, 2019) (Pradhan et al., 2019) (Yadav and Dutta, 2020) (Someoliayi et al., 2019) (Tran et al., 2020) (Azizi and Do, 2018a) (Azizi and Do, 2018b) (Hilton et al., 2018) (Chen et al., 2018) (Marcozzi et al., 2017) (Akour et al., 2018) (Öztürk, 2017) (Arrieta et al., 2018) (Singh and Srivastava, 2018) (Andrianto et al., 2018)	22
4	Repositori Siemens / SIR repository	(R. Wang et al., 2020) (Su et al., 2020) (Mondal and Nasre, 2021) (Pachariya, 2020) (Huang et al., 2020b) (Ba-quttayyan et al., 2019) (Özener and Sözer, 2020) (Rahmani et al., 2021) (Khatibsyarbini et al., 2019) (Shrivathsan et al., 2019) (Silva et al., 2019) (Di Nucci et al., 2020) (Miranda et al., 2018) (Lin et al., 2018) (Choudhary et al., 2018) (Khatibsyarbini et al., 2017) (Fu et al., 2018) (Bertolino et al., 2017) (Wongwuttiwat and Lawanna, 2018)	19
5	Repositori Defect4J	(Mahdieh et al., 2020) (Venugopal et al., 2020) (Chi et al., 2020) (Ba-quttayyan et al., 2019) (Rahmani et al., 2021) (Wang et al., 2018) (Abdur et al., 2018) (Shin et al., 2019)	8
6	Lain-lain	(Jahan et al., 2020) (Al-Sabbagh et al., 2020) (Sun et al., 2020) (Awasthi et al., 2020) (Taneja et al., 2020) (Halim et al., 2019) (Zhang et al., 2019) (Khanna et al., 2019b) (Ji et al., 2019) (Vescan et al., 2017) (Nayak et al., 2017) (Lun et al., 2017) (Ali et al., 2017)	13

SUT dari Open Source

Untuk menjawab RQ2 (Bagaimana karakteristik open source untuk SUT yang digunakan para peneliti *software testing*), hasil identifikasi menunjukkan beberapa informasi penting.

Github merupakan sumber *open-source* yang dimanfaatkan para peneliti untuk membangun SUT. Secara umum peneliti melakukan penanaman *fault* pada *software*, dengan skenario sesuai tujuan penelitian. Selanjutnya dibuat serangkaian *test-case* (*test suite*) yang disiapkan untuk "menjaring" *fault* yang telah ditanam. Dalam beberapa SUT, para peneliti menggunakan sistem versioning untuk memperoleh beragam SUT dari suatu program. Setiap versi SUT biasanya dibedakan dengan jumlah dan variasi *fault*. Dengan demikian jumlah *test case* pun berbeda untuk setiap versi. Tabel 4 memperlihatkan artikel yang menggunakan *open source* sebagai SUT.

Tabel 4. Daftar Penelitian Menggunakan Open Source sebagai SUT

Artikel	Deskripsi Open Source
(Azizi and Do, 2018a)	Aplikasi *Open-Source e-commerce* berbasis *website* bernama nopCommerce
(Chen et al., 2018)	Sistem Open-Source Apache HBase
(Arrieta et al., 2018)	Sistem Open Source untuk mengontrol DC Engine
(Huang et al., 2020a)	5 versi dari 5 program Open-Source GNU FTP Server
(H. Wang et al., 2020)	8 aplikasi BPEL (Business Process Execution Language): Travel, ATM, GymLocker, Marketplace, Auction, Risk, Assestment, Loan
(Eldrandaly et al., 2020)	Aplikasi Fee Report dengan bahasa Java

(Mohd-Shafie et al., 2020)	Tiga aplikasi Open Source: Online Jewellery Shopping, Car Rental System, dan Blood Bank Management System
(Li et al., 2020)	12 aplikasi open source dari eclipse community: jdt core, jdt debug, jdt ui, pde ui, platform debug, platform resources, platform runtime, platform team, platform text, platform ua, dan platform ui, dan equinox framework. Serta menggunakan 6 aplikasi open source dar apache community: Apache OOZIE, Apache ratis, Apache Streams, Beam, Apache Hadoop, dan Apache HBase
(Rahmani et al., 2021)	Aplikasi Open Source OpenMRS, NumbertoWord, dan Palindrom
(Butool et al., 2019)	Aplikasi Open-Source ATM, Online Food Ordering, dan Process Flow Manager
(Hettiarachchi and Do, 2019)	Aplikasi iTrust versi 0, 1, 2, dan 3
(Pradhan et al., 2019)	Dua versi produk VCS dari Cisco, serta aplikasi Paint Control dan IOF/ROL dari ABB Robotic
(Yadav and Dutta, 2020)	Aplikasi Open-Source ATM
(Someoliayi et al., 2019)	Aplikasi Apache Commons Math dan Apache Commons Lang
(Tran et al., 2020)	Aplikasi Open Source BugZilla, Launchpad, Mantis, dan Debian
(Azizi and Do, 2018b)	Aplikasi Open Source nopCommerce, Umbraco-CMS, jmeter, jtopas
(Hilton et al., 2018)	47 Proyek Open-Source yang dibangun dengan Bahasa Java, Python, Elixer, Scala, Golang, Node.js, dan DotNet
(Marcozzi et al., 2017)	14 Aplikasi Open Source menggunakan Bahasa C: 7 dari program Siemens, 4 dari cryptographic OpenSSL toolkit, 1 GNU Zip Compression Program, 1 aplikasi Sjeng Chess, dan SQLite relational database management system
(Akour et al., 2018)	Aplikasi Open-Source Cinema
(Öztürk, 2017)	5 aplikasi open source: Data-structures-csharp, Epicylce.math, GimSharp, ProjectEuler, dan YAMP
(Singh and Srivastava, 2018)	Aplikasi Open-Source AspectJ dan HealthWatcher

Repositori SUT

Software infrastructure repository (SIR) dan Defect4J merupakan database SUT yang bersifat *open source.* Keduanya merupakan repositori SUT yang sangat populer. Dapat dilihat dari 84 artikel terpilih, 27 di antaranya menggunakan SIR, atau sekitar 32% sebagai SUT.

Hingga saat ini SIR menjadi repositori SUT yang paling banyak digunakan. Sejumlah SUT pada SIR telah tersedia secara lengkap dan banyak diminati para peneliti khususnya untuk program berbahasa C. Dari hasil identifikasi SUT pada tabel 3, dapat dilihat jumlah penelitian yang memanfaatkan program-program pada SIR cukup tinggi. Informasi mengenai SUT pada SIR dikupas cukup detail pada studi sebelumnya (Min et al., 2021). Tabel 5 adalah beberapa SUT berbahasa C pada SIR.

Tabel 5. Beberapa SUT pada SIR

Nama Program	Deskripsi
printtokens	lexical analyzers
printtokens2	lexical analyzers
replace	pattern replace
schedule	priority scheduler
schedule2	priority scheduler
tcas	altitude separation
totinfo	information measure
space	Array Definition Language (ADL) interpreter
bash	shell script interpreter
flex	pattern matching
grep	pattern matching
gzip	file compressor/expander
make	makefile commands executor
sed	stream editor for filtering and transforming text
vim	text editor

Pengkategorian objek SUT SIR sebelumnya pernah dilakukan oleh Khatibsyahbini (2018), yang secara tersirat menyebutkan SUT SIR dalam bahasa C dibagi menjadi tiga yaitu Siemens, Space, dan TSL (Khatibsyarbini et al., 2018). Nama "TSL" digunakan karena objek memerlukan tools TSL untuk dieksekusi. Program yang dimaksudkan adalah flex, grep, gzip, dan make. Namun, nama Unix Utilities lebih tepat karena objek SUT tersebut adalah command unix utilities dan diambil dari situs GNU oleh SIR ("SIR Repository," n.d.). Organisasi objek-objek SUT pada SIR dibedakan menjadi organisasi objek "lama" (untuk program Siemens dan Space) dan organisasi objek "baru" (selain Siemens dan Space). Lebih detail mengenai SUT pada SIR dapat dilihat pada Tabel 6.

Tabel 6. Karakteristik SUT pada SIR

Nama Program	Tipe fault	Jumlah versi	Jumlah baris	Jumlah test case	Jumlah Fault	Jumlah Prosedure	Ukuran (Mb)
printtokens	Seeded	1	726	4130	7	18	122
printtokens2	Seeded	1	570	4115	10	19	95
replace	Seeded	1	564	5542	32	21	203
schedule	Seeded	1	412	2650	9	18	50
schedule2	Seeded	1	374	2710	10	16	55
tcas	Seeded	1	173	1608	41	9	24
totinfo	Seeded	1	565	1052	23	7	33
space	Real	1	6199	13585	38	136	1454
bash	Seeded	7	59846	1061	32	1061	79
flex	Seeded	6	10459	570	81	162	14
grep	Seeded	6	10068	809	57	146	13
gzip	Seeded	6	5680	217	59	104	18
make	Seeded	5	35545	1043	35	268	261
sed	Real, Seeded	8	14427	370	32	255	10
vim	Seeded	8	122169	975	22	1999	163

Seeded: jenis fault yang sengaja dibuat; real: jenis fault sesungguhnya

Di samping SIR, SUT repositori yang bersifat *open source* lainnya adalah Defet4J. Repositori Defect4J berisi sejumlah SUT dalam Bahasa Java, terlihat dari namanya Defect4J yang berarti Defect for Java. Dari 86 artikel pada Tabel 3, terdapat 8 peneliti yang menggunakan SUT dari Defect4J.

Defects4J merupakan *database* yang menyediakan *bugs* yang berasal dari proyek-proyek *open source* berbahasa Java. Selain itu, Defects4J juga menyediakan infrastruktur pembantu untuk menggunakan *bugs-bugs* tersebut sehingga dapat digunakan untuk eksperimen.

Sebagai contoh, berdasarkan studi (Gay and Just, 2020), Defects4J telah digunakan untuk mengevaluasi *formatted test generation*, *program repair* otomatis, dan *fault localization research*. Pada versi Defects4J saat ini, yaitu versi 2.0.0 (Defect4J, n.d.), terdapat total 835 *bugs* yang berasal dari 17 proyek *open source* dengan bahasa Java. Tabel 7 adalah karakteristik dari proyek *open source* yang digunakan oleh Defects4J.

Tabel 7. Karakteristik Proyek pada Defects4J

No	Identifier	Nama Proyek	Jumlah Bug	ID bug	Bug tidak digunakan
1	Chart	jfreechart	26	1 - 26	None
2	Cli	commons-cli	39	1 - 5, 7 - 40	6
3	Closure	closure-compiler	174	1 - 62, 64 - 92, 94 - 176	63, 93
4	Codec	commons-codec	18	1 - 18	None
5	Collections	commons-collection	4	25 - 28	1 – 24
6	Compress	commons-compress	47	1 - 47	None
7	Csv	commons-csv	16	1 - 16	None
8	Gson	gson	18	1 - 18	None
9	JacksonCore	jackson-core	26	1 - 26	None
10	JacksonDataBind	jackson-databind	112	1 - 112	None
11	JacksonXml	jackson-dataformat-xml	6	1 - 6	None
12	Jsoup	jsoup	93	1 - 93	None
13	JxPath	commons-jxpath	22	1 - 22	None
14	Lang	commons-lang	64	1, 3 - 65	2
15	Math	commons-math	106	1 - 106	None
16	Mockito	mockito	38	1 - 38	None
17	Time	joda-time	26	1 - 20, 22 - 27	21

Setiap *bug* pada Defects4J memiliki karakteristik berikut (Gay and Just, 2020):

1. meliputi versi *source code* yang *buggy* dan versi *source code* yang *fixed*. Versi *source code* yang *fixed* merupakan suatu *source code* yang telah di-*fix* dari suatu *issue* yang ditemukan oleh *project's issue tracker*. Perubahan yang dilakukan oleh versi *fixed* ini hanya dilakukan terhadap *source code*, bukan terhadap artefak dari proyek seperti *config* atau *build files*.
2. merupakan *reproducible*: pada versi *source code* yang *fixed*, semua *test* akan berjalan dengan sukses, sedangkan pada versi *buggy*, akan terdapat salah satu *test* yang gagal untuk mengekspos *bug*-nya.
3. setiap *bug* diisolasi: perbedaan antara versi *source code* yang *fixed* dan *buggy* dibuat seminimal mungkin, dan semuanya terkait dengan *bug* yang bersangkutan. Dengan kata lain, perbedaan antara versi *fixed* dan versi *buggy* terbebaskan dari perubahan *code* yang tidak terkait dengan *bug* yang bersangkutan, seperti *refactoring* atau penambahan fitur.

Untuk setiap *bug*, Defects4J menyediakan artefak dan metadata berikut:

1. Sepasang versi *source code* - versi *buggy* dan versi *fixed*
2. Sekumpulan *class* dan *source code* yang termodifikasi oleh *patch* yang digunakan untuk membenarkan *bug* yang ada.
3. Sekumpulan *developer-written tests* yang dapat mengekspos *bug* - disebut dengan *"trigger tests"*
4. Sebuah *stacktrace* untuk setiap *trigger tests*, ketika *test* tersebut dieksekusi di versi *buggy*
5. Sekumpulan *class* yang di-*load* oleh *classloader* selama *trigger tests* dieksekusi.
6. Sekumpulan *test* yang berhubungan dengan *bug* tertentu

IV. PENUTUP

Kesimpulan

SUT sangat penting dalam proses penelitian software testing. Untuk hal tersebut kajian mengenai SUT sangat bermanfaat. Dari 84 artikel yang dikaji, terlihat penggunaan beberapa SUT cukup tinggi. Beberapa SUT dibangun dari software yang tersebar pada platform Github sebagai open source. Untuk pemanfaatan SUT dari repositori, SIR dan Defect4J merupakan repositori SUT yang paling banyak digunakan oleh para peneliti.

Batasan SLR

Salah satu aspek penting pada penelitian berupa SLR adalah adanya keterbatasan tertentu. Dalam hal ini, kemungkinan proses pencarian artikel yang kurang sempurna, sehingga sumber artikel yang menjadi dasar studi memiliki keterbatasan. Hal ini mengingat ranah *software testing* begitu luas. Berbagai *keyword* yang telah digunakan, sangat mungkin belum sepenuhnya dapat "menjaring" seluruh artikel dari sumber-sumber yang semestinya dapat dijaring.

Saran-saran

SLR mengenai SUT dapat dieksplorasi lebih jauh, dengan meningkatkan kuantitas maupun kualitas artikel. Di samping itu, parameter dapat lebih diperkaya, dengan menentukan variable khusus sesuai tujuan SLR. Misalnya, apakah pada setiap jenis teknik *testing*, digunakan SUT yang memiliki karakteristik khusus.

Ucapan Terima Kasih

Terima kasih disampaikan kepada P3M Politeknik Negeri Bandung, yang telah mendukung dan membiayai penelitian ini.

DAFTAR PUSTAKA

Abbas, M., Inayat, I., Saadatmand, M., Jan, N., 2019. Requirements dependencies-based test case prioritization for extra-functional properties. Proc. - 2019 IEEE 12th Int. Conf. Softw. Testing, Verif. Valid. Work. ICSTW 2019 159–163.

Abdur, M., Abu, M., Saeed, M., 2018. Prioritizing Dissimilar Test Cases in Regression Testing using Historical Failure Data. Int. J. Comput. Appl. 180, 1–8.

Akour, M., Abuwardih, L., Alhindawi, N., Alshboul, A., 2018. Test Case Minimization using Genetic Algorithm: Pilot Study. 2018 8th Int. Conf. Comput. Sci. Inf. Technol. CSIT 2018 66–70.

Al-Sabbagh, K.W., Staron, M., Ochodek, M., Hebig, R., Meding, W., 2020. Selective Regression Testing based on Big Data: Comparing Feature Extraction Techniques. Proc. - 2020 IEEE 13th Int. Conf. Softw. Testing, Verif. Valid. Work. ICSTW 2020 322–329.

Alfieri, A., Castiglione, C., Pastore, E., 2020. A multi-objective tabu search algorithm for product portfolio selection: A case study in the automotive industry. Comput. Ind. Eng. 142, 106382.

Ali, S., Li, Y., Yue, T., Zhang, M., 2017. An

empirical evaluation of mutation and crossover operators for multi-objective uncertainty-wise test minimization. Proc. - 2017 IEEE/ACM 10th Int. Work. Search-Based Softw. Testing, SBST 2017 21–27.

Anderson, J., Azizi, M., Salem, S., Do, H., 2019. On the use of usage patterns from telemetry data for test case prioritization. Inf. Softw. Technol. 113, 110–130.

Andrianto, I., Liem, M.M.I., Asnar, Y.D.W., 2018. Web application fuzz testing. Proc. 2017 Int. Conf. Data Softw. Eng. ICoDSE 2017 2018-Janua, 1–6.

Arrieta, A., Wang, S., Markiegi, U., Sagardui, G., Etxeberria, L., 2018. Employing Multi-Objective Search to Enhance Reactive Test Case Generation and Prioritization for Testing Industrial Cyber-Physical Systems. IEEE Trans. Ind. Informatics 14, 1055–1066.

Awasthi, U., Palmer, K.A., Bollas, G.M., 2020. Optimal test and sensor selection for active fault diagnosis using integer programming. J. Process Control 92, 202–211.

Azizi, M., Do, H., 2018a. A collaborative filtering recommender system for test case prioritization in web applications. Proc. ACM Symp. Appl. Comput. 1560–1567.

Azizi, M., Do, H., 2018b. Graphite: A Greedy Graph-Based Technique for Regression Test Case Prioritization. Proc. - 29th IEEE Int. Symp. Softw. Reliab. Eng. Work. ISSREW 2018 245–251.

Ba-quttayyan, B., Mohd, H., Yusof, Y., 2019. Regression Test Case Prioritization Frameworks: Challenges and Future Directions. Int. J. Recent Technol. Eng. 8, 8457–8462.

Bertolino, A., Miranda, B., Pietrantuono, R., Russo, S., 2017. Adaptive Coverage and Operational Profile-Based Testing for Reliability Improvement. Proc. - 2017 IEEE/ACM 39th Int. Conf. Softw. Eng. ICSE 2017 541–551.

Bosmans, S., Mercelis, S., Denil, J., Hellinckx, P., 2019. Testing IoT systems using a hybrid simulation based testing approach. Computing 101, 857–872.

Butool, R., Nadeem, A., Sindhu, M., Zaman, O.U., 2019. Improving requirements coverage in test case prioritization for regression testing. Proc. - 22nd Int. Multitopic Conf. INMIC 2019.

Chen, B., Song, J., Xu, P., Hu, X., Jiang, Z.M., 2018. An automated approach to estimating code coverage measures via execution logs. ASE 2018 - Proc. 33rd ACM/IEEE Int. Conf. Autom. Softw. Eng. 305–316.

Chi, J., Qu, Y., Zheng, Q., Yang, Z., Jin, W., Cui, D., Liu, T., 2020. Relation-based test case prioritization for regression testing. J. Syst. Softw. 163.

Choudhary, A., Agrawal, A.P., Kaur, A., 2018. An effective approach for regression test case selection using pareto based multi-objective harmony search. Proc. - Int. Conf. Softw. Eng. 13–20.

Chuaychoo, N., Kansomkeat, S., 2017. Path coverage test case generation using genetic algorithms. J. Telecommun. Electron. Comput. Eng. 9, 115–119.

Dadkhah, M., Araban, S., Paydar, S., 2020. A systematic literature review on semantic web enabled software testing. J. Syst. Softw. 162, 110485.

Defect4J, n.d. Defect4J [WWW Document].

Di Nucci, D., Panichella, A., Zaidman, A., De Lucia, A., 2020. A Test Case Prioritization Genetic Algorithm Guided by the Hypervolume Indicator. IEEE Trans. Softw. Eng. 46, 674–696.

Eldrandaly, K., Ellatif, M.A., Zaki, N., 2020. A proposed framework for test suite prioritization and reduction using the clustering data mining technique. J. Theor. Appl. Inf. Technol. 98, 290–307.

Fu, W., Yu, H., Fan, G., Ji, X., Pei, X., 2018. A Regression Test Case Prioritization Algorithm Based on Program Changes and Method Invocation Relationship. Proc. - Asia-Pacific Softw. Eng. Conf. APSEC 2017-Decem, 169–178.

Garousi, V., Felderer, M., Kılıçaslan, F.N., 2019. A survey on software testability. Inf. Softw. Technol. 108, 35–64.

Gay, G., Just, R., 2020. Defects4J as a Challenge Case for the Search-Based Software Engineering Community. Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics) 12420 LNCS, 255–261.

Hajri, I., Goknil, A., Pastore, F., Briand, L.C., 2020. Automating system test case classification and prioritization for use case-driven testing in product lines. Empir. Softw. Eng. 25, 3711–3769.

Halim, S.A., Jawawi, D.N.A., Sahak, M., 2019. Similarity distance measure and prioritization algorithm for test case prioritization in software product line testing. J. Inf. Commun. Technol. 18, 57–75.

Hasnain, M., Pasha, M.F., Ghani, I., Jeong, S.R., 2021. Functional Requirement-Based Test Case Prioritization in Regression Testing: A Systematic Literature Review, SN Computer Science. Springer Singapore.

Hettiarachchi, C., Do, H., 2019. A Systematic Requirements and Risks-Based Test Case Prioritization Using a Fuzzy Expert System. Proc. - 19th IEEE Int. Conf. Softw. Qual. Reliab. Secur. QRS 2019 374–385.

Hilton, M., Bell, J., Marinov, D., 2018. A large-scale study of test coverage evolution. ASE 2018 - Proc. 33rd ACM/IEEE Int. Conf. Autom. Softw. Eng. 53–63.

Huang, R., Sun, W., Chen, T.Y., Towey, D., Chen, J., Zong, W., Zhou, Y., 2020a. Abstract Test Case Prioritization Using Repeated Small-Strength Level-Combination Coverage. IEEE Trans. Reliab. 69, 349–372.

Huang, R., Zhang, Q., Towey, D., Sun, W., Chen, J., 2020b. Regression test case prioritization by code combinations coverage. J. Syst. Softw. 169, 110712.

Hynninen, T., Kasurinen, J., Knutas, A., Taipale, O., 2018. Software testing: Survey of the industry practices. 2018 41st Int. Conv. Inf. Commun. Technol. Electron. Microelectron. MIPRO 2018 - Proc. 1449–1454.

Jahan, H., Feng, Z., Mahmud, S.M.H., 2020. Risk-Based Test Case Prioritization by Correlating System Methods and Their Associated Risks. Arab. J. Sci. Eng. 45, 6125–6138.

Ji, S., Li, B., Zhang, P., 2019. Test Case Selection for All-Uses Criterion-Based Regression Testing of Composite Service. IEEE Access 7, 174438–174464.

Khanna, M., Chaudhary, A., Toofani, A., Pawar, A., 2019a. Performance Comparison of Multi-objective Algorithms for Test Case Prioritization During Web Application Testing. Arab. J. Sci. Eng. 44, 9599–9625.

Khanna, M., Chauhan, N., Sharma, D.K., 2019b. Search for prioritized test cases during web application testing. Int. J. Appl. Metaheuristic Comput. 10, 1–26.

Khatibsyarbini, M., Isa, M.A., Jawawi, D.N.A., 2017. A hybrid weight-based and string distances using particle swarm optimization for prioritizing test cases. J. Theor. Appl. Inf. Technol. 95, 2723–2732.

Khatibsyarbini, M., Isa, M.A., Jawawi, D.N.A., Hamed, H.N.A., Mohamed Suffian, M.D., 2019. Test Case Prioritization Using Firefly Algorithm for Software Testing. IEEE Access 7, 132360–132373.

Khatibsyarbini, M., Isa, M.A., Jawawi, D.N.A., Tumeng, R., 2018. Test case prioritization approaches in regression testing: A systematic literature review. Inf. Softw. Technol. 93, 74–93.

Kitchenham, B., Brereton, P., 2013. A systematic review of systematic review process research in software engineering. Inf. Softw. Technol. 55, 2049–2075.

Lei Xiao, Huaikou, Weiwei Zhuang, S.C., 2017. An Empirical Study on Clustering Approach Combining Fault Prediction for Test Case Prioritization. In: 2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS). pp. 815–820.

Li, Y., Wang, J., Yang, Y., Wang, Q., 2020. An extensive study of class-level and method-level test case selection for continuous integration. J. Syst. Softw. 167, 110614.

Lin, J.W., Jabbarvand, R., Garcia, J., Malek, S., 2018. Nemo: Multi-criteria test-suite minimization with integer nonlinear programming. Proc. - Int. Conf. Softw. Eng. 1039–1049.

Lun, L., Wang, S., Chi, X., Xu, H., 2017. Automatic generation of basis component path coverage for software architecture testing. Comput. Informatics 36, 386–404.

Luo, L., 2000. Software Testing Techniques. In: Software Testing Techniques Technology Maturation and Research Strategy. Institute for Software Research International, Carnegie Mellon University.

Magalhães, C., Andrade, J., Perrusi, L., Mota, A., Barros, F., Maia, E., 2020. HSP: A hybrid selection and prioritisation of regression test cases based on information retrieval and code coverage applied on an industrial case study. J. Syst. Softw. 159.

Mahdieh, M., Mirian-Hosseinabadi, S.H., Etemadi, K., Nosrati, A., Jalali, S., 2020. Incorporating fault-proneness estimations into coverage-based test case prioritization methods. Inf. Softw. Technol. 121, 106269.

Marcozzi, M., Bardin, S., Kosmatov, N., Papadakis, M., Prevosto, V., Correnson, L., 2017. Freeing Testers from Polluting Test Objectives.

Matinnejad, R., Nejati, S., Briand, L.C., 2017. Automated testing of hybrid simulink/stateflow controllers: Industrial case studies. Proc. ACM SIGSOFT Symp. Found. Softw. Eng. Part F1301, 938–943.

Matinnejad, R., Nejati, S., Briand, L.C., Bruckmann, T., 2019. Test Generation and

Test Prioritization for Simulink Models with Dynamic Behavior. IEEE Trans. Softw. Eng. 45, 919–944.

Min, J.L., Rajabi, N., Rahmani, A., 2021. Comprehensive study of SIR: Leading SUT repository for software testing. J. Phys. Conf. Ser. 1869.

Miranda, B., Cruciani, E., Verdecchia, R., Bertolino, A., 2018. FAST approaches to scalable similarity-based test case prioritization. Proc. - Int. Conf. Softw. Eng. 2018-Janua, 222–232.

Mishra, D.B., Mishra, R., Acharya, A.A., Das, K.N., 2019. Test case optimization and prioritization based on multi-objective genetic algorithm, Advances in Intelligent Systems and Computing. Springer Singapore.

Mohd-Shafie, M.L., Wan-Kadir, W.M.N., Khatibsyarbini, M., Isa, M.A., 2020. Model-based test case prioritization using selective and even-spread count-based methods with scrutinized ordering criterion. PLoS One 15, 1–27.

Mondal, S., Nasre, R., 2021. Summary of Hansie: Hybrid and consensus regression test prioritization. Proc. - 2021 IEEE 14th Int. Conf. Softw. Testing, Verif. Validation, ICST 2021 278–280.

Nayak, S., Kumar, C., Tripathi, S., 2017. Enhancing Efficiency of the Test Case Prioritization Technique by Improving the Rate of Fault Detection. Arab. J. Sci. Eng. 42, 3307–3323.

Ouriques, J.F.S., Cartaxo, E.G., Machado, P.D.L., 2018. Test case prioritization techniques for model-based testing: a replicated study. Softw. Qual. J. 26, 1451–1482.

Özener, O.Ö., Sözer, H., 2020. An effective formulation of the multi-criteria test suite minimization problem. J. Syst. Softw. 168, 110632.

Öztürk, M.M., 2017. A bat-inspired algorithm for prioritizing test cases. Vietnam J. Comput. Sci.

Pachariya, M.K., 2020. Building Ant System for Multi-Faceted Test Case Prioritization: An Empirical Study. Int. J. Softw. Innov. 8, 23–37.

Pradhan, D., Wang, S., Ali, S., Yue, T., Liaaen, M., 2019. Employing rule mining and multi-objective search for dynamic test case prioritization. J. Syst. Softw. 153, 86–104.

Rahmani, A., Min, J.L., Maspupah, A., 2020. An Evaluation of Code Coverage Adequacy in Automatic Testing using Control Flow Graph Visualization. ISCAIE 2020 - IEEE 10th Symp. Comput. Appl. Ind. Electron. 239–244.

Rahmani, A., Min, J.L., Maspupah, A., 2021. An empirical study of regression testing techniques. J. Phys. Conf. Ser. 1869.

Riskiawan, H.Y., Azhari, 2017. Automated software testing system using multi-agent system characteristics approach. Adv. Sci. Lett. 23, 2389–2391.

Samad, A., Mahdin, H., Kazmi, R., Ibrahim, R., 2021. Regression Test Case Prioritization: A Systematic Literature Review. Int. J. Adv. Comput. Sci. Appl. 12, 655–663.

Shin, D., Yoo, S., Papadakis, M., Bae, D.H., 2019. Empirical evaluation of mutation-based test case prioritization techniques. Softw. Test. Verif. Reliab. 29, 1–28.

Shin, K.W., Lim, D.J., 2020. Model-based test case prioritization using an alternating variable method for regression testing of a UML-based model. Appl. Sci. 10, 1–23.

Shrivathsan, A.D., Ravichandran, K.S., Krishankumar, R., Sangeetha, V., Kar, S., Ziemba, P., Jankowski, J., 2019. Novel fuzzy clustering methods for test case prioritization in Software Projects. Symmetry (Basel). 11, 1–22.

Silva, D.S., Rabelo, R., Neto, P.S., Britto, R., Oliveira, P.A., 2019. A test case prioritization approach based on software component metrics. Conf. Proc. - IEEE Int. Conf. Syst. Man Cybern. 2019-Octob, 2939–2945.

Sinaga, A.M., 2017. Case study on testing of web-based application: Del's students information system. J. Telecommun. Electron. Comput. Eng. 9, 1–5.

Singh, M., Srivastava, V.M., 2018. Extended firm mutation testing: A cost reduction technique for mutation testing. 2017 4th Int. Conf. Image Inf. Process. ICIIP 2017 2018-Janua, 604–609.

SIR Repository [WWW Document], n.d.

Sobreira, V., Durieux, T., Madeiral, F., Monperrus, M., De Almeida Maia, M., 2018. Dissection of a bug dataset: Anatomy of 395 patches from Defects4J. 25th IEEE Int. Conf. Softw. Anal. Evol. Reengineering, SANER 2018 - Proc. 2018-March, 130–140.

Someoliayi, K.E., Jalali, S., Mahdieh, M., Mirian-Hosseinabadi, S.H., 2019. Program State Coverage: A Test Coverage Metric Based on Executed Program States. SANER 2019 - Proc. 2019 IEEE 26th Int. Conf. Softw. Anal. Evol. Reengineering 584–588.

Su, W., Li, Z., Wang, Z., Yang, D., 2020. A Meta-

heuristic Test Case Prioritization Method Based on Hybrid Model. Proc. - 2020 Int. Conf. Comput. Eng. Appl. ICCEA 2020 430–435.

Sun, Z., Zhang, J.M., Harman, M., Papadakis, M., Zhang, L., 2020. Automatic testing and improvement of machine translation. Proc. - Int. Conf. Softw. Eng. 974–985.

Taneja, D., Singh, R., Singh, A., Malik, H., 2020. A Novel technique for test case minimization in object oriented testing. Procedia Comput. Sci. 167, 2221–2228.

Tran, H.M., Le, S.T., Nguyen, S. Van, Ho, P.T., 2020. An Analysis of Software Bug Reports Using Machine Learning Techniques. SN Comput. Sci. 1.

Ulewicz, S., Vogel-heuser, B., Member, S., 2018. Prioritization in Production Automation 1–13.

Vasanthapriyan, S., Tian, J., Zhao, D., Xiong, S., Xiang, J., 2017. An ontology-based knowledge management system for software testing. Proc. Int. Conf. Softw. Eng. Knowl. Eng. SEKE 230–235.

Venugopal, Y., Quang-Ngoc, P., Eunseok, L., 2020. Modification point aware test prioritization and sampling to improve patch validation in automatic program repair. Appl. Sci. 10, 1–14.

Vescan, A., Şerban, C., Chisăliţă-Creţu, C., Dioşan, L., 2017. Requirement dependencies-based formal approach for test case prioritization in regression testing. Proc. - 2017 IEEE 13th Int. Conf. Intell. Comput. Commun. Process. ICCP 2017 181–188.

Wang, H., Yang, M., Jiang, L., Xing, J., Yang, Q., Yan, F., 2020. Test Case Prioritization for Service-Oriented Workflow Applications: A Perspective of Modification Impact Analysis. IEEE Access 8, 101260–101273.

Wang, R., Li, Z., Jiang, S., Tao, C., 2020. Regression Test Case Prioritization Based on Fixed Size Candidate Set ART Algorithm. Int. J. Softw. Eng. Knowl. Eng. 30, 291–320.

Wang, Y., Zhu, Z., Yang, B., Guo, F., Yu, H., 2018. Using reliability risk analysis to prioritize test cases. J. Syst. Softw. 139, 14–31.

Wongwuttiwat, J., Lawanna, A., 2018. Performance improvement model of regression test selection. ICIIBMS 2017 - 2nd Int. Conf. Intell. Informatics Biomed. Sci. 2018-Janua, 49–53.

Yadav, D.K., Dutta, S., 2020. Regression test case selection and prioritization for object oriented software. Microsyst. Technol. 26, 1463–1477.

Yu, Z., Fahid, F., Menzies, T., Rothermel, G., Patrick, K., Cherian, S., 2019. TERMINATOR: Better automated UI test case prioritization. ESEC/FSE 2019 - Proc. 2019 27th ACM Jt. Meet. Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng. 883–894.

Zhang, T., Wang, X., Wei, D., Fang, J., 2018. Test Case Prioritization Technique Based on Error Probability and Severity of UML Models 28, 831–844.

Zhang, W., Qi, Y., Zhang, X., Wei, B., Zhang, M., Dou, Z., 2019. On test case prioritization using ant colony optimization algorithm. Proc. - 21st IEEE Int. Conf. High Perform. Comput. Commun. 17th IEEE Int. Conf. Smart City 5th IEEE Int. Conf. Data Sci. Syst. HPCC/SmartCity/DSS 2019 2767–2773.