Survei Pengukuran Fleksibilitas *Software* Menggunakan Metode *Systematic Literature Review*

Junaedi Abdillah[1,*], Dimas Fahmi Rizaldi[2], Machmud Naufal[3], Muhammad Ainul Yaqin[4], Abd. Charis Fauzan[5]

[1,2,3,4]Program Studi Teknik Informatika, Universitas Islam Negeri Maulana Malik Ibrahim, Indonesia
[5]Program Studi Ilmu Komputer, Universitas Nahdlatul Ulama Blitar, Indonesia

[1]19650070@student.uin-malang.ac.id; [2]19650067@student.uin-malang.ac.id; [3]19650068@student.uin-malang.ac.id; [4]yaqinov@ti.uin-malang.ac.id; [5]abdcharis@unublitar.ac.id

*corresponding author

INFO ARTIKEL

Sejarah Artikel
Diterima: 18 Juni 2021
Direvisi: 5 Juli 2021
Diterbitkan: 30 April 2022

Kata Kunci
Survei pengukuran *Software*
Pengukuran Fleksibilitas
Fleksibilitas *Software*
Metode perhitungan fleksibilitas

ABSTRAK

Fleksibilitas menjadi faktor penting yang harus dipertimbangkan ketika pembuatan *software*. Namun pembahasan pengukuran fleksibilitas *software* masih cukup baru. Menyebabkan keterbatasan dalam pemahaman pengukuran fleksibilitas software. Tujuan penelitian ini untuk menjabarkan berbagai macam metode pengukuran fleksibilitas *software* dan memperoleh metode pengukuran terbaik. *Paper* ini berisi hasil SLR (*Systematic Literature Review*) mengenai pembahasan pengukuran fleksibilitas *software* yang meliputi aspek, metode, dan hal lain yang perlu dipertimbangkan untuk pengukuran. Hasil dari penelitian ini adalah temuan 4 model pengukuran fleksibilitas yang dibagi menjadi dua perspektif, serta penentuan metode terbaik untuk pengukuran fleksibilitas software. Kesimpulan yang didapatkan adalah metode pengukuran fleksibilitas didasarkan pada *Flexible Points* oleh Limin Shen paling baik untuk mengukur fleksibilitas *software*.

PENDAHULUAN

Fleksibilitas memiliki beberapa pengertian namun secara umum memiliki definisi yang sama yaitu kemampuan untuk beradaptasi terhadap setiap perubahan yang terjadi[1]. Pengukuran fleksibilitas dapat diartikan pengumpulan informasi untuk mengetahui besaran atau nilai kuantitatif tingkat fleksibilitas suatu objek. Fleksibilitas *software* adalah "kemudahan untuk memodifikasi sebuah sistem atau komponen suntuk digunakan dalam sebuah aplikasi yang sebelumnya telah dirancang secara khusus [2]." Dalam pengertian lain fleksibilitas *software* merupakan salah satu sifat yang menunjukkan jika sebuah perangkat lunak mudah di ubah dan dapat beradaptasi pada suatu kondisi [3]. Pengukuran terhadap fleksibilitas *software* dapat dibagi menjadi dua perspektif yaitu *software process flexibility* dan *software product flexibility*[4]. *Software process flexibility* merupakan perspektif fleksibilitas *software* yang bertujuan untuk mengukur kemampuan *software* dalam menyesuaikan dan mengikuti perubahan *environment* ataupun proses bisnis [4], kemampuan ini penting diterapkan untuk membuat *software* yang bisa menyesuaikan tingginya perubahan pada proses bisnis. *Software product flexibility* mengacu pada kemampuan *software* untuk dimodifikasi dan dirubah struktur internalnya dari kondisi satu ke kondisi lain dengan mudah untuk mencapai desain *software* yang diinginkan [4].

Perubahan yang sering terhadap model proses bisnis ataupun tuntutan perubahan internal pada sistem untuk meningkatkan kualitas atau kemampuan suatu *software* menyebabkan isu fleksibilitas menjadi penting untuk dibahas[1]. Selain itu, kondisi tersebut menyadarkan para pengembang tentang pentingnya aspek fleksibilitas dalam mendesain sistem sehingga level

fleksibilitasnya mampu ditingkatkan seiring berjalannya waktu[1]. Namun pembahasan mengenai aspek pengukuran fleksibilitas *software* ataupun model perhitungan secara kuantitatif masih sangat terbatas [1]. Hal ini karena luasnya cangkupan yang harus dipertimbangkan ketika mengukur fleksibilitas *software*. Dan juga sulitnya menemukan variabel yang dapat digunakan untuk mengukur fleksibilitas yang nantinya dapat digunakan dalam perbandingan fleksibilitas antar *software* kedepannya [3].

Tujuan dari penelitian menentukan metode pengukuran paling baik dalam mengukur fleksibilitas *software* sehingga membantu proses analisis untuk tingkatan fleksibilitas suatu *software* untuk meningkatkan fleksibilitas. Oleh karena itu, Penelitian ini berisi rangkuman metode pengukuran berdasarkan dua perspektif utama yaitu *Software process flexibility* dan *Software product flexibility*. Metode pengukuran dirumuskan berdasarkan aspek fleksibilitas yang relevean. Metode penelitian yang digunakan adalah *Systematic Literature Review* (SLR) yang diharapkan dengan adanya penelitian ini terdapat keterbaharuan dalam metode pengukuran fleksibilitas *software* serta meningkatnya pemahaman tentang pengukuran fleksibilitas *software*.

METODE

Pada studi ini digunakan metode SLR. Tinjau pustaka sistematis adalah sebutan lain dari metode SLR (*Systematic Literatur Review*). Identifikasi, Penilaian, dan Interpretasi dari seluruh yang ditemukan pada suatu topik penelitian adalah sebuah inti dari metode SLR. Metode ini digunakan untuk menjawab pertanyaan penelitian (*research question*) yang telah ditetapkan sebelumnya [5]. Tahapan pada SLR dibagi menjadi 3 yaitu *Planning*, *Conducting*, dan *Reporting*. Pada Tahap planning atau perencanaan, memuat beberapa pertanyaan dalam penelitian menurut PICOC (*Population Intervetion Comparison Outcome Context*). Tahap berikutnya adalah conduction atau pelaksanaan yang memuat tentang penentuan *keyword*, penentuan sumber literatur,dan pembuatan kriteria inklusi dan ekclusi, yang terakhir adalah reporting atau pelaporan, yang berisi tentang laporan hasil pencarian literatur

.

Tahap *Planning*

Research Question (RQ) adalah awalan dari adanya metode SLR. Pencarian literatur didasari dari RQ yang sudah ada, dan jawaban dari RQ pada metode SLR adalah analisis dan sintesis. Ada beberapa kriteria yang harus memenuhi untuk mendapatkan RQ yang sesuai, yakni bermanfaat, terstruktur, dan mengarah ke *step-of-the-art-research* dari suatu topic penelitian. RQ yang dibuat didasarkan pada 5 elemen fundamental yang disebut PICOC (*Population Intervention Comparison Outcomes Context*) yang dijabarkan pada Tabel 1.

Tabel 1. Tabel Penentuan RQ

PICOC	Kriteria	Penelitian
P	*Population* (target dari penelitian)	Perspektif dalam pengukuran fleksibilitas *software*
I	*Intervention* (aspek detail dari penelitian)	Aspek setiap perspektif dalam mengukur fleksibilitas *software*
C	*Comparison* (membandingkan hasil *intervention*)	Membandingkan aspek paling relevan dari setiap perspektif untuk membuat model perhitungan secara kuantitatif
O	*Outcomes* (hasil dari penelitian)	Model perhitungan secara kuantitatif untuk mengukur fleksibilitas *software*.
C	*Context* (lingkungan dari penelitian)	Di bidang akademik dan bisnis yang memerlukan pengukuran fleksibilitas *software*

Berdasarkan tabel 1 kami merumuskan beberapa RQ (*Research Question*), antara lain

- RQ1 : Apa yang meningkatkan kebutuhan akan fleksibilitas sebuah *software*?
- RQ2 : Apa saja perspektif dalam pengukuran fleksibilitas *software*?
- RQ3 : Aspek apa saja yang terdapat dalam setiap perspektif untuk mengukur fleksibilitas suatu *software*?
- RQ4 : Apa saja metode pengukuran berdasarkan aspek yang ditemukan?
- RQ5 : Metode apa yang paling baik untuk mengukur fleksibilitas *software*?

Tahap *Conducting*

Awal dari sebuah tahapan ini adalah dengan menentukan adanya *keyword* dari pencarian literatur yang berbasis pada PICOC. *Keyword* sinonim yang ada pada penelitian ini adalah sebagai berikut:
("Pengukuran" atau "*Measurement*" atau "Metrics" atau "*Evaluation*")
dan
("Fleksibilitas" atau "*Flexibility*" atau "*Adaptability*")
dan
("*Software*" atau "Perangkat Lunak")

Setelah semua keyword sudah didapat maka langkah berikutnya adalah dengan menelusuri dan menentukan sumber dari pencarian literatur. Pada Tabel 2 kami merangkum hasil penelusuran dari literatur yang kami gunakan.

Tabel 2. Tabel Penelusuran dan Penentuan Sumber Pencarian Literatur

Perpustakaan Digital	Halaman Web	Batasan Keyword
IEEE Xplore (IEEE)	ieeexplore.ieee.org	Memiliki judul yang relevan terhadap penelitian
Research Gate	researchgate.net	Memiliki judul yang relevan terhadap penelitian
Journal of Soft Computing and Information Technology (JSCIT)	jscit.nit.ac.ir	Memiliki judul yang relevan terhadap penelitian
Science Direct	sciencedirect.com/	Memiliki judul yang relevan terhadap penelitian
IET Digital Library	digital-library.theiet.org/	Memiliki judul yang relevan terhadap penelitian
Springer link	link.springer.com	Memiliki judul yang relevan dengan topic pembahasan
JAIS e-Library	aisel.aisnet.org	Memiliki judul yang relevan terhadap topic pembahasan
ACM Digital Library	dl.acm.org	Memiliki judul yang relevan terhadap penelitian
Jurnal Teknologi Informasi dan Ilmu Komputer	jtiik.ub.ac.id	Memiliki judul yang relevan terhadap penelitian

Selanjutnya hanya literatur yang sesuai yang digunakan. Dengan tujuan untuk memudahkan proses penyeleksian, maka pemilihan literatur dilakukan dengan membentuk kriteria yang meliputi kriteria inklusi dan eksklusi. Kriteria tersebut kami rangkum kedalam Tabel 3.

Tabel 3. Tabel Pengelompokan Jurnal yang Sesuai

Kriteria Inklusi	Kriteria Eksklusi
Membahas mengenai pengukuran fleksibilitas *software*	Tidak membahas implementasi perhitungan fleksibilitas *software*

Membahas mengenai ciri-ciri pengukuran fleksibilitas *software*	Pembahasan ciri ciri kurang lengkap
Membahas aspek dalam pengukuran fleksibilitas *software*	Tidak menjelaskan aspek pengukuran secara mendetail
Membahas metode untuk mengukur fleksibilitas *software*	Tidak menjelaskan detail dari Perhitungan fleksibilitas *software*

Tahap *Reporting*

Dari temuan hasil yang diperoleh, *paper* yang relevan dikategorikan menjadi dua klasifikasi besar berdasarkan topik yang ingin dibahas yaitu pengukuran fleksibilitas dalam perspektif *software process flexibility* dan *software product flexibility* (lihat Tabel 4).

Tabel 4. Tabel Pengklasifikasian Jurnal

No.	Perspektif Pengukuran	Paper yang Relevan
1	*Software Product Flexibility*	[3] [6] [7] [8] [9] [10] [11]
2	*Software Process Flexibility*	[1] [4] [9] [10] [11] [12] [13] [14] [15] [16] [17]

Kebaruan Ilmiah

Sebagai kontribusi ilmiah serta pembeda dengan hasil penelitian yang ada sebelumnya, kami menyajikan metode metode pengukuran fleksibilitas software menjadi dua perspektif yaitu *software process flexibility* dan *software product flexibility*. Berdasarkan perspektif tersebut kami mengumpulkan aspek fleksibilitas yang relevan untuk menyajikan metode pengukuran berdasarkan aspek tersebut. Kami juga menyajikan hasil analisis metode yang paling baik untuk mengukur suatu fleksibilitas *software* (lihat tabel 11).

PEMBAHASAN

Faktor yang Meningkatkan Kebutuhan Fleksibilitas *software* (RQ1)

Cepatnya perubahan yang terjadi dalam proses bisnis dan perkembangan *software* menyebabkan isu pengukuran fleksibilitas penting untuk dikaji[1]. Dan faktor lain seperti perubahan kepentingan dan target pasar serta berbagai faktor lain menuntut suatu organisasi atau perusahaan untuk memiliki system *software* yang fleksibel. *Paper* [14] menjelaskan terdapat 6 faktor yang meningkatkan kebutuhan fleksibilitas software, yaitu :

a. Perkembangan teknologi

   Kebutuhan mengenai fleksibilitas *software* dapat didorong karena perubahan dari dukungan infrastruktur *hardware* dan *software*. Contohnya sistem game yang harus diinstall di internal komputer bisa digantikan dengan gaming yang berbasis *cloud*, yang semuanya terintegrasi menjadi satu serta client tidak memerlukan lagi *hardware* yang *powerful*.

b. *Exceptions* (kesalahan atau faktor yang belum dipertimbangkan)

   Ketika *software* telah dibuat bisa terjadi *use-case-scenario* dari user yang tidak diperhitungkan ataupun faktor lain yang tidak dipertimbangkan. Hal ini bisa menyebabkan ketidakstabilan sistem atau mengancam keamanan. *Software* harus bisa dibuat dengan desain jika terjadi kesalahan atau error maka harus ada "Exit Strategy" untuk melanjutkan proses yang ada.

c. Penggunaan metode pengembangan baru

   Perubahan strategi organisasi untuk mencapai tujuan menyebabkan perubahan metode pengembangan *software*. Sebagai contoh perubahan strategi perusahaan *publisher* game untuk memperoleh pasar *console*, sedangkan game yang telah dibuat

hanya bisa dijalankan pada perangkat PC (*Personal Computer*). Dari contoh ini diperlukan metode baru untuk pengembangan agar bisa menjalankan game pada kedua perangkat.

d. Perubahan target tujuan

Perubahan target organisasi mengharuskan *software* mampu beradaptasi dengan proses *maintenance* se-minimal mungkin. Perubahan target bisa terjadi karena faktor perubahan pasar ataupun fokus dalam pengembangan suatu *software* yang dilakukan organisasi.

e. Perubahan aturan pemerintah atau lisensi hukum

Suatu perubahan aturan atau kebijakan mengharuskan *software* mampu beradaptasi dengan baik. Sebagai contoh perubahan kebijakan *privacy* mengharuskan *software* mengikuti aturan baru yang telah ditetapkan.

f. Penghematan

Faktor terakhir adalah pertimbangan alasan ekonomi, artinya untuk menghemat uang ataupun waktu dalam pelayanan. *Software* diharuskan mampu beradaptasi dengan perubahan sedikit atau biaya semurah mungkin.

Perspektif dalam Pengukuran Fleksibilitas *Software* (RQ2)

Pengukuran terhadap fleksibilitas *software* dapat dibagi menjadi dua perspektif yaitu *software process flexibility* dan *software product flexibility*[4]. *Software process flexibility* merupakan perspektif fleksibilitas *software* yang bertujuan untuk mengukur kemampuan *software* dalam menyesuaikan dan mengikuti perubahan *environment* ataupun proses bisnis [4], kemampuan ini penting diterapkan untuk membuat *software* yang bisa menyesuaikan tingginya perubahan pada proses bisnis. *Software product flexibility* mengacu pada kemampuan *software* untuk dimodifikasi dan dirubah struktur internalnya dari kondisi satu ke kondisi lain dengan mudah untuk mencapai desain *software* yang diinginkan [4].

Pengukuran Fleksibilitas *Software* dalam Perspektif *Software process Flexibility*

Software process flexibility merupakan perspektif fleksibilitas *software* yang bertujuan untuk mengukur kemampuan *software* dalam menyesuaikan dan mengikuti perubahan *environment* ataupun proses bisnis [4]. Untuk menjawab RQ3 dan mengetahui aspek yang diperlukan dalam mengukur fleksibiltas berdasarkan perspektif ini, kami merangkum aspek fleksibilitas berdasarkan penelitian terkait kedalam Tabel 5.

Tabel 5. Tabel Aspek Fleksibilitas Perspektif *Software process Flexibility*

No	Penelitian	Aspek Fleksibilitas
1	Gong, Yiwei Janssen, Marijn	- *Extent* - *Duration* - *Swiftness* - *Anticipation*
2	Afflerbach, Patrick Kastner, Gregor Krause, Felix Röglinger, Maximilian	- *Process Flexibility*
3	Cognini, Riccardo Corradini, Flavio Gnesi, Stefania Polini, Andrea Re, Barbara	- *Variability* - *Adaptation* - *Loosenesss* - *Evolution* - *Strategy and Organizational Flexibility* - *Manufacturing Flexibility*

		- *Mechanism Flexibility*
4	Gebauer, Judith Schober, Franz	- IS IT *Flexibility* - *Variability* - *Time Criticality* - *Strategies Flexibility* - *Flexibility by Use* - *Flexibility by Change* - Economic
5	Pamungkas, Endang Wahyu Sinaga, Fernandes Rochimah, Siti	- *Manufacturing Flexibility* - *Process flexibility* - *Cost* - *Handling Flexibility*
6	Rosemann, Michael Recker, Jan	- *Adaptation* - *Modification* - *Process design*
7	Subramaniam, Hema Zulzalil, Hazura	- *Process Model* - *Management and Concept*

Berdasarkan Tabel 5, terdapat 2 metode pengukuran fleksibilitas *software* yang relevan yang selanjutnya akan menjawab RQ4 yaitu metode model metrik yang diperkenalkan oleh E. Pamungkas dalam *paper* [1] dan model pengukuran yang diperkenalkan oleh J. Gebauer dalam *paper* [12].

A. Model Metrik untuk Mengukur Fleksibilitas oleh E. Pamungkas

Model pengukuran ini digunakan untuk mengukur tingkat fleksibilitas model suatu bisnis proses, namun karena bisnis proses secara langsung mempengaruhi fleksibilitas *software* dalam perspektif *Software process Flexibility* sehingga kami putuskan untuk mencantumkan model pengukuran ke dalam *paper* yang kami buat.

Pengukuran fleksibilitas menggunakan model metrik pada *paper* [1] dilakukan dengan membuat aspek fleksibilitas menjadi model metrik. Pada tahap awal pembuatan model disajikan aspek fleksibilitas yang akan digunakan kemudian dijabarkan pada tabel 6.

Tabel 6. Tabel Aspek Fleksibilitas yang Digunakan pada Model Metrik

No	Aspek Fleksibilitas	Pengertian
1	*Range*	Kemampuan untuk memudahkan implementasi perubahan dari satu kondisi ke dalam kondisi lain
2	*Cost*	Fleksibilitas terjadi jika usaha atau *effort* yang diperlukan untuk melakukan perubahan rendah
3	*Time*	Model yang fleksibel jika wkatu yang diperlukan melakukan perubahan singkat atau dengan kata lain efisien
4	*Paralellism*	Kemampuan pengerjaan *task* atau tugas secara pararel
5	*Choice*	Kemampuan memilih tingkat urgensi *task* yang ada untuk dikerjakan
6	*Interleaving*	Kemampuan eksekusi *task* secara urut tanpa ada eksekusi secara bersamaan
7	*Extensible*	Kemampuan mengimplementasi penambahan aktivitas lain
8	*Reduce*	Kemampuan mengimplementasi pengurangan dari model yang telah ada
9	*Relink*	Kemampuan membuat *edge* yang menghubungkan struktur model
10	*Creation*	Kemampuan menangani penambahan *task* kedalam model
11	*Delegation*	Kemampuan menangani perpindahan data dari *task* yang satu ke *task* lain
12	*Nesting*	Kemampuan menangani sub-proses secara bercabang

Aspek pada Tabel 6 kemudian dibagi menjadi beberapa level berdasarkan tingkat urgensi aspek tersebut terhadap kebutuhan akan fleksibilitas *software*. Dari 12 aspek yang dijabarkan pada tabel 6, dibagi menjadi 4 level, level pertama yang memiliki kepentingan

lebih tinggi akan diberi bobot penilaian lebih tinggi dan juga seterusnya. Dari nilai bobot yang didapat dijumlahkan untuk mendapat derajat fleksibilitas. Hasil pengelompokan ditulis dalam tabel 7 yang merupakan model metric untuk mengukur fleksibilitas.

Tabel 7. Pengelompokan Aspek Fleksibilitas pada Model Metric

Tingkat	Aspek
Level 1	- Range - *Cost* - Time
Level 2	- Extensible - Reduce - Relink
Level 3	- Creation - Delegation - Nesting
Level 4	- Paralellism - Choice - Interleaving

Pembagian level pada Tabel 7 menentukan pembobotan dalam perhitungan nilai fleksibilitas. Sehingga level teratas lebih menentukan *software* dikategorikan fleksibel atau tidak. Setiap aspek fleksibilitas yang didukung dalam *software* maka diberi nilai satu, yang selanjutnya menghasilkan *metric* perhitungan pada Persamaan (1)

$$Flexibility = \frac{4\ x\ Leve1{+}3\ x\ Level2{+}2\ x\ Level3{+}\ 1\ x\ Level4}{30} \qquad (1)$$

Kesimpulan perhitungan dari persamaan (1), model proses bisnis dikatakan fleksibel jika hasil pengukuran mendekati 1. Ketika model bisnis proses dikatakan fleksibel, maka *software* yang dibuat seharusnya fleksibel. Hal ini sejalan, karena model proses bisnis yang baik akan menghasilkan *software* berkualitas yang membuat *software* dapat menangani berbagai perubahan yang terjadi dan akan terjadi. Dari proses bisnis yang fleksibel akan memiliki berbagai *exception* terhadap masalah yang belum atau sudah diprediksi, yang akan menghemat *cost* dan time dalam implementasi perubahan kepada *software*.

B. Model Pengukuran Fleksibilitas oleh J. Gebauer

Pada model pengukuran ini difokuskan untuk mengidentifikasi strategi fleksibilitas yang akan mempengaruhi *cost efficiency* dari proses bisnis tertentu. Pengukuran didasarkan pada tiga aspek fleksibilitas utama yaitu *flexibility to use* (w1), *flexibility to change* (w2), dan *manual performance of process tasks* (w3). Pengukuran ini juga didasarkan pada tiga masalah utama yaitu *Uncertainty Effect* (p), *Variability Effect* (v), dan *Time Criticality* (r). Dari permasalahan tersebut dibagi menjadi dua *stage* untuk perancangan penyelesaian masalah yaitu *stage* 1 (t0) dan *stage* 2 (t1). Stage 1 berisi perancangan keputusan mengenai *cost flexibility to use* (*ICOST*) dan *cost* of *flexibility to change* (*FCOST*). Kemudian terdapat satu variabel y yaitu variabel untuk menampung nilai apakah *software* memiliki dukungan *flexibility to change* atau tidak. Stage 2 berisi perancangan keputusan tentang process task terhadap *cost* dari setiap proses yang dilakukan seperti System Upgrade (*UCOST*), System Operating (*OCOST*), dan Manual Operating (*MCOST*).

Dari aspek dan permasalahan yang ada dibuat menjadi model pengukuran. Berikut ini model pengukuran serta penjelasan dari beberapa variabel yang digunakan.dalam model pengukuran yang diperkenalkan oleh J. Gebauer berdasarkan aspek dan permasalah yang telah disebutkan sebelumnya.

1) w1 = p x1
   w2 = (1-p) x2
   w3 = p (1-x1) + (1-p) (1-x2) = 1 – w1 – w2
   Dimana "x1" dan "x2" merupakan variabel tingkat penggunaan sistem. x1 adalah nilai antisipasi terhadap t0. Dan x2 nilai antisipasi terhadap t1.
2) $y \geq x2$
   Nilai x2 hanya bisa positif jika *flexibility to change* tersedia dalam t0.
3) *ICOST* = a + b L(x1)
   ICOST adalah *cost* untuk mengimplementasi *flexibility to use*. "a" adalah komponen biaya tetap yang terpisah untuk melakukan tugas tertentu yang berhubungan dengan *Information System* seperti pengaturan struktur umum database dan User Interface, penyediaan fungsionalitas sistem dasar dan kapasitas pemrosesan. "b" adalah *cost* untuk melakukan antisipasi terhadap t0. Dan L(x1) adalah pembagian tugas potensial yang termasuk ke dalam *Information System*, dimana x1 sesuai dengan nilai kurva Lorenz.
4) *FCOST* = c y
   "c" adalah nilai komponen biaya tetap terkait penyediaan sumber daya yang memadai, integritas data dan fungsionalitas, serta modularitas suatu komponen sistem.
5) *OCOST* = d (w1 + w2)
   "d" adalah perkiraan biaya operasi selama berjalannya sistem dari proses bisnis yang dilakukan menggunakan *information system*.
6) *UCOST* = e L(x2)
   UCOST adalah *cost* dari *upgrade* sistem untuk aktivitas proses yang tidak diantisipasi. L(x2) mengindikasikan titik pada kurva lorenz yang sesuai dengan pengerjaan upgrade sistem. Dan "e" mengambarkan *upgrade cost* yang dibutuhkan untuk semua proses yang tidak diantisipasi dalam t0 namun diketahui berdasar informasi pada t1.
7) *MCOST* = f (1+r g) w3
   "f" variabel biaya operasi jika semua proses dilakukan manual dalam t1. Dan "r" mengambarkan pembagian *time critical activities*, dimana "g" mengindikasikan *cost markup* untuk time critical activities.
8) *TCOST* = *ICOST* + *FCOST* + *OCOST* + *UCOST* + *MCOST*
   Dimana hasil dari *TCOST* mempengaruhi tingkat efisiensi dan fleksibilitas *software*. Ketika perhitungan *TCOST* menghasilkan angka kecil, bisa dikatakan *software* tersebut fleksibel dan memiliki *cost* rendah dalam melakukan berbagai proses. Yang mana dalam pembahasan *paper* [12] dilanjutkan cara untuk meningkatkan efisiensi dan cara menekan *cost*, namun karena pembahasan tersebut tidak termasuk fokus kami sehingga tidak dicantumkan.

Pengukuran Fleksibilitas *Software* dalam Perspektif *Software product Flexibility*

Software product flexibility mengacu pada kemampuan *software* untuk dimodifikasi dan dirubah struktur internalnya dari kondisi satu ke kondisi lain dengan mudah untuk [4]. Pertimbangan fleksibilitas *software* berdasarkan *software product flexibility* memiliki cangkupan yang cukup luas dimulai dari arsitektur *software* yang dibuat, paradigma pemrograman yang digunakan apakah menggunakan OOP (*Object Oriented Programming*) atau *procedural programming*, *design pattern*, serta cangkupan lain [7]. Masing masing

cangkupan tersebut menentukan kemudahan dalam perubahan internal suatu *software* yang secara langsung akan mempengaruhi kefleksibilitasannya.

Untuk mengukur fleksibiltas berdasarkan perspektif *software product flexibility* dan menjawab RQ3, kami merangkum aspek terkait berdasarkan penelitian yang relevan kedalam Tabel 8.

Tabel 8. Tabel Aspek Pengukuran Fleksibilitas Perspektif *Software product Flexibility*

No.	Penelitian	Aspek Fleksibilitas
1	Eden, A. H. Mens, T.	- *Programming pardigms* - *Architectural Styles* - *Design patterns*
2	Shen, Limin Ren, Shangping	- *Software Flexibility* - Architecture *Flexibility* - Time - *Cost*
3	Rasoolzadegan, Abbas	- *Structure Design* - *Programming paradigms* - *Architecture Software* - *Design pattern*
4	Davis, Daniel	- *Programming paradigms* - *Logic Programming*

Dari Tabel 8, terdapat 2 metode pengukuran fleksibilitas *software* yang relevan yang selanjutnya akan menjawab RQ4 yaitu *Evolution cost metrics* yang diperkenalkan oleh Eden, A. H. dalam *paper* [7] dan Metode pengukuran fleksibilitas didasarkan pada *Flexible Points* yang diperkenalkan oleh Limin Shen dalam *paper* [3].

A. *Evolution cost metrics*

Metode ini akan mengukur *cost* dari "*evolution process*" menggunakan notasi dari "*evolution cost metric*". *Cost* yang didapat dari perhitungan ini bisa mengindikasikan ukuran fleksibilitas *software* jika terjadi perubahan komponen atau lainnya. Perubahan yang terjadi bisa berupa penambahan, pengurangan, atau perubahan modul pada *software* sebagai hasil dari perkembangan *software*. Nilai perubahan diperoleh dengan menghitung perbedaan *symmetric set* antara modul lama (i_{old}) dengan modul yang telah diperbaharui ($i_{adjusted}$). Formulasi lengkap perhitungan mengenai *Evolution Cost Metrics* dapat dilihat pada Tabel 9.

Tabel 9. Formulasi Perhitungan *Evolution cost metrics*

Evolution cost metric ($C^1_{Classes}$) akan menghitung *cost* dari implementasi yang dilakukan pada *evolution step* $\varepsilon=<<P_{old}, P_{shifted}, i_{old}>, i_{adjusted}>$ dalam kategori yang dipengaruhi oleh perubahan yang terjadi,

$$C^1_{Classes}(\varepsilon) \triangleq |\, Classes\,(i_{old}, i_{adjusted})|$$

Dimana $\Delta Classes\,(i_{old}, i_{adjusted})$ menunjukkan perbedaan *symmetric set* antara kelas/elemen pada program, yaitu

$$(Classes(i_{old}) \setminus Classes(i_{shifted}) \cup Classes(i_{shifted}) \setminus Classes(i_{old}))$$

Keterangan :

P = *Problem* (Permasalahan)

i = *Implementation* (Penerapan)

Formulasi pada Tabel 9 dapat digunakan untuk mengukur tingkat fleksibilitas suatu *software*. Sebagai contoh permasalahan, mengutip dari *paper* [7] tentang penilaian fleksibilitas *Java Collection interface*, dimana data struktur (ε_{DS}) pada *interface* ini fleksibel karena ketika dilakukan perubahan *cost of executing* (ε_{DS}) tidak bergantung kepada jumlah data struktur, sedangkan untuk penambahan operasi (ε_{OP}) pada *interface* ini tidak fleksibel karena *cost of executing* (ε_{OP}) linier dengan penambahan jumlah data struktur. Dari permasalahan ini dilakukan pemodelan pengukuran dengan *evolution cost metric*, misalkan $C^1_{Classes}$ dapat digunakan untuk mengukur *cost* dari perubahan struktur data (ε_{DS}) dan *cost* dari penambahan operasi (ε_{OP}). Dan anggap bahwa program komputer melakukan *evolution process* seperti program sedang menyesuaikan implementasi yang telah ada. Kompleksitas dari permasalahan ini dihitung menggunakan *evolution cost metric* $C^1_{Classes}$ dengan pemodelan

1. $O(C^1_{Classes}(\varepsilon_{DS})) =$
   O (# kelas/elemen program yang terkena dampak perubahan *ArrayList* dengan *LinkedList*) =
   O (1)
2. $O(C^1_{Classes}(\varepsilon_{OP})) =$
   O (# kelas/elemen program yang terkena dampak oleh penambahan operasi *remove* pada *library* "*Collection*) =
   O (|DS|)

Kompleksitas dari setiap *evolution step* dapat diringkas seperti pada tabel 10.

Tabel 10. Tabel Kompleksitas dari *Evolution Step*

Evolution Step / Design Police	Change Data Structure (ε_{DS})	Add Operation (ε_{OP})
Collection interface	O(1)	O(|DS|)

Keterangan :

- O(1) : Notasi *Big* O *Constant Time*
- O(n) : Notasi *Big* O *Linier Time*

Model diatas diperoleh karena pada kasus permasalahan yang digunakan *cost* dari perubahan data struktur (ε_{DS}) pada *Java Collection Interface* adalah konstan dan tidak bergantung pada implementasi yang terjadi, sedangkan operasi (ε_{OP}) pada *interface* ini tidak fleksibel karena *cost of executing* (ε_{OP}) sejalan dengan implementasi yang dilakukan. Dari hasil ini menguatkan bahwa klaim *Collection design policy* tidak fleksibel dalam absolute terms, namun hasil ini juga menunjukkan bahwa *Java Collection Interface* fleksibel terhadap penambahan data struktur dan tidak fleksibel terhadap penambahan operasi.

Dari contoh permasalahan diatas, dapat dilihat bahwa penggunaan metode pengukuran ini lebih kepada analisis *cost* yang dibutuhkan dalam melakukan perubahan sebelum dan sesudah implementasi. Untuk memperoleh *cost* dibutuhkan model perhitungan, dimana pembuatan model persamaan bisa menyesuaikan dari permasalahan yang diangkat, nantinya akan mempengaruhi bentuk metrik dan model persamaan. Dari model tersebut dianalisis *cost* yang dibutuhkan, kemudian digunakan untuk menentukan suatu *software* fleksibel atau tidak.

B. Metode pengukuran fleksibilitas didasarkan pada *Flexible Points*

Metode ini digunakan untuk mengukur secara kuantitatif setiap *point* fleksibilitas. Pada pembahasan awal metode pengukuran dengan *Flexible Points* adalah memperkenalkan "Dorongan/*Force*" yang menyebabkan perubahan pada *software*. "Dorongan/*Force*" *F* terdiri dari *external force* F_e dan *internal force* F_i dimana $F = F_e + F_i$. *Internal Force* F_i adalah dorongan didalam *software* itu sendiri, misalkan sistem *software* dapat menyediakan *adaptive interface* yang memungkinkan user memperoleh kapasistas untuk melakukan suatu konfigurasi atau personalisasi, perubahan *design pattern*, atau perubahan *architecture* style. *External Force* F_e adalah dorongan yang berasal dari interaksi *software* dengan *environtment*, seperti user yang memanipulasi *software*, seorang *maintainer* atau *developer* yang memodifikasi *software*, sebuah aplikasi yang mengkontrol input *stream*, dan perubahan *environtment* pada *software*.

Selanjutnya dalam *paper* [3] memperkenalkan kuantitatif konsep dan metrik berdasarkan *software flexibility evaluation* sebagai berikut :

- *FleXible Point (FXP) i* : merupakan *point* dalam *software* yang dapat menyebabkan perubahan fleksibilitas, dimana *external force* F_e dapat berpengaruh. F_e bisa menyebabkan perubahan pada *software* melalui *flexible point*. Sedikit *external force* F_e pada FXP bisa menyebabkan pengaruh yang besar terhadap perubahan *software*. Ketika F_e=0, itu mengindikasikan bahwa perubahan yang terjadi disebabkan oleh *internal force* F_i.
- *Flexible Force* F_i : adalah indikasi tingkat kesulitan dalam melakukan perubahan perangkat lunak. Jika nilai F_i besar maka akan sulit melakukan perubahan melalui FXP i.
- *Flexible Distance* S_i : adalah *range* atau ukuran maksimum suatu *software* dapat melakukan perubahan berdasarkan F_i melalui *flexible point* i.
- *Flexible Degree* K_i : $K_i = S_i / (1 + F_i)$: adalah ukuran dari fleksibilitas *software* pada FXP i.
- *Flexible Capacity* C : $C = \sum_{i=1}^{N} Ki$: adalah ukuran keseluruhan atau sebagian dari fleksibilitas *software*.
- Berdasarkan definisi diatas, seorang manipulator dapat memanfaatkan fleksibilitas pada i hanya jika $F_e \geq F_i$.

Berdasarkan *paper* [3] terdapat pengolongan untuk *Flexible point* menjadi 5 jenis yang berbeda, antara lain:

- *Potential* FXP (PFXP) : semua *flexible point* yang mungkin digunakan dibawah pengaturan *environment* yang sesuai. *Capacity* of PFXP, $C_{PFXP} = \sum_{i=1}^{N} Ki$
- *Available* FXP (AFXP) : *flexible point* yang mana user memiliki kemampuan untuk melakukan manipulasi. *Capacity of* AFXP, $C_{AFXP} = \sum_{i=1}^{N} Ki \mid F_e \geq f_i$
- *User* FXP (UFXP) : *flexible point* yang telah dimanipulasi dan digunakan oleh *user*. UFXP berarti bahwa *users* telah menggunakan *Flexible Points*, *software* telah berubah, beberapa *flexible mechanisms* telah aktif, dan efek fleksibilitas telah terlihat dalam *software behaviours. Capacity of* UFXP, $C_{UFXP} = \sum_{i=1}^{N} K(i) \mid F_e \geq f_i$
- *Current* FXP (CFXP) : *Flexible Points* yang dapat dimanipulasi oleh users. *Flexible capacity of* CFXP, $C = \sum_{i=1}^{N} K(i) \mid F_e \geq f_i$
- *Required* FXP (RFXP) : *Flexible Points* yang tidak ada dalam *software* namun dibutuhkan untuk ditambahkan kedalam *software*.

Dari persamaan diatas dalam paper [3] terdapat beberapa kesimpulan antara lain,

- *Rate of* FXP *availability* (RA) = jumlah dari AFXP atau jumlah dari PFXP. Jika RA bernilai rendah, itu berarti manipulator atau *developer* tidak cocok dan harus disesuaikan.

- *Rate of* FXP *suitability* (RS) = jumlah dari UFXP atau jumlah dari PFXP. Jika RS bernilai rendah, itu berarti beberapa *software* FXP tidak sesuai atau banyak FXP baru dibutuhkan atau manipulator/*developer* tidak kompeten dalam pengoperasian, atau perubahan sedikit *requirement*. Jika RS bernilai rendah dan jumlah RFXP bernilai lebih tinggi daripada jumlah PFXP, dapat diartikan desain FXP tidak efektif, dan terdapat masalah serius pada desain FXP.
- Dari pernyataan diatas, dapat disimpulkan bahwa jika nilai RA dan RS yang rendah mengindikasikan suatu *software* memiliki masalah pada *Flexible Point* (FXP) yang diartikan *software* tersebut memiliki fleksibilitas yang buruk.

Analisis untuk Menentukan Model Pengukuran paling Baik (RQ5)

Tabel 11. Tabel Analisis Penentuan Model Pengukuran

Perspektif Pengukuran	Model Pengukuran	Kelebihan	Kekurangan
Software process Flexibility	Model Metrik untuk Mengukur Fleksibilitas *Software* oleh E. Pamungkas	+ Untuk mengukur fleksibilitas proses bisnis, model ini memiliki cangkupan pengukuran yang luas dari aspek yang diambil. Sehingga hasil yang didapat menggambarkan secara penuh kondisi dari model proses bisnis yang dihitung.	- Untuk mengukur fleksibilitas *software*, model ini memiliki kekurangan karena hanya berfokus kepada perhitungan model proses bisnis. Pada model pengukuran ini tidak dibahas korelasi fleksibilitas *software* yang dikembangkan dengan fleksibilitas model proses bisnis.
	Model Pengukuran Fleksibilitas oleh J. Gebauer	+ Model pengukuran cocok untuk menghitung fleksibilitas dalam lingkup strategi fleksibilitas yang akan mempengaruhi *cost efficiency* dari proses bisnis tertentu. Fleksibilitas *software* dalam model pengukuran ini dipandang dari seberapa besar *cost* yang diperlukan ketika ada perubahan, perawatan, pengunaan sumber daya, dan lainnya.	- Hasil pengukuran yang didapat sebatas seberapa besar "*cost*" yang dihasilkan ketika ada aktivitas yang dilakukan.
Software product Flexibility	*Evolution cost metrics* oleh Eden, A. H.	+ Model pengukuran mudah diimplementasikan untuk menganalisis fleksibilitas suatu *software* pada perspektif *software product flexibility* terhadap perubahan yang dilakukan seperti mengukur fleksibilitas *software* berdasarkan *design pattern*, architecture style, atau programming paradigm.	- Hasil pengukuran yang didapat adalah nilai fleksibilitas *software* secara kualitatif, karena hasil berupa analisis seberapa besar fleksibilitas struktur *software* terhadap perubahan yang terjadi.
	Metode pengukuran fleksibilitas didasarkan pada *Flexible Points* oleh Limin Shen	+ Hasil dari model pengukuran ini berupa data secara kualitatif dan kuantitatif. Analisis didasarkan pada perhitungan flexible point dimana terdapat potensi perubahan secara internal ataupun eskternal pada *software*.	- Kekurangan metode ini mungkin hanya minor yaitu tidak menjelaskan secara jelas bagaimana memperoleh ukuran maksimum suatu *software* dapat melakukan perubahan dalam kuantitatif konsep Flexible Distance.

Dari Tabel 11 dapat disimpulkan jika metode pengukuran fleksibilitas didasarkan pada *Flexible Points* oleh Limin Shen merupakan metode yang paling baik untuk mengukur fleksibilitas *software* terhadap berbagai perubahan. Karena cakupan aspek yang diukur dalam metode ini cukup luas meliputi faktor eksternal dan internal pada *software*. Kemudian keterkaitan aspek yang digunakan dengan fleksibilitas *software* sangat relevan. Hasil yang didapat dari pengukuran ini cukup detail berupa data kualitatif dan kuantitatif.

KESIMPULAN

Pengukuran fleksibilitas *software* penting dilakukan dalam pengembangan *software*, agar fleksibilitas yang dimiliki dapat ditingkatkan sehingga mampu mengikuti perubahan terhadap model proses bisnis ataupun tuntutan perubahan internal pada sistem. Dalam Pengukuran *software* fleksibilitas dibagi menjadi dua perspektif yaitu *software process flexibility* dan *software product flexibility*. *Software process flexibility* merupakan perspektif fleksibilitas *software* yang bertujuan untuk mengukur kemampuan *software* dalam menyesuaikan dan mengikuti perubahan environment ataupun proses bisnis. Dan *software product flexibility* mengacu pada kemampuan *software* untuk dimodifikasi dan dirubah struktur internalnya dari kondisi satu ke kondisi lain dengan mudah untuk mencapai desain *software* yang diinginkan. Untuk meneliti tentang pengukuran fleksibilitas *software* kami menggunakan metode *Systematic Literatur Review* (SLR). SLR terdiri dari 3 bagian yaitu *Planning*, *Conducting*, dan *Reporting*. Dari penelitian menggunakan SLR kami memperoleh kesimpulan bahwa metode pengukuran fleksibilitas didasarkan pada *Flexible Points* oleh Limin Shen yang paling baik digunakan untuk mengukur fleksibilitas *software*, karena metode pengukuran ini memiliki cangkupan pengukuran aspek fleksibilitas *software* yang luas meliputi faktor eksternal dan internal pada *software*. Kemudian dengan aspek yang digunakan dalam perhitungan sangat relevan dengan fleksibilitas *software*, hasil yang diperoleh juga cukup detail berupa data kualitatif dan kuantitatif.

REFERENSI

[1] E. W. Pamungkas, F. Sinaga, and S. Rochimah, "Model Metric untuk Mengukur Fleksibilitas Model Proses Bisnis," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 1, no. 2, p. 62, 2014, doi: 10.25126/jtiik.201412113.

[2] Ieee, "IEEE Standard Glossary of Software Engineering Terminology," *Office*, vol. 121990, no. 1, p. 1, 1990, doi: 10.1109/IEEESTD.1990.101064.

[3] L. Shen and S. Ren, "Analysis and measurement of software flexibility based on flexible points," *3th Softw. Meas. Eur. Forum, Italy*, no. March, pp. 331–341, 2006.

[4] H. Subramaniam and H. Zulzalil, "Software quality assessment using flexibility: A systematic literature review," *Int. Rev. Comput. Softw.*, vol. 7, no. 5, pp. 2095–2099, 2012.

[5] B. Kitchenham, "Source: 'Guidelines for performing Systematic Literature Reviews in SE', Kitchenham et al Guidelines for performing Systematic Literature Reviews in Software Engineering," 2007.

[6] D. Davis, "Modelled on Software Engineering: Flexible Parametric Models in the Practice of Architecture," no. February, 2013.

[7] A. H. Eden and T. Mens, "Measuring software flexibility," *IEE Proc. Softw.*, vol. 153, no. 3, pp. 113–125, 2006, doi: 10.1049/ip-sen:20050045.

[8] A. Rasoolzadegan, "A new approach to the quantitative measurement of software reliability," *J. Inf. Syst. Telecommun.*, vol. 3, no. 3, pp. 165–172, 2015, doi: 10.7508/jist.2015.03.005.

[9] D. D. Zeng and J. L. Zhao, "Achieving software flexibility via intelligent workflow techniques," *Proc. Annu. Hawaii Int. Conf. Syst. Sci.*, vol. 2002-Janua, no. February 2002, pp. 606–615, 2002, doi: 10.1109/HICSS.2002.993941.

[10] K. M. Nelson, H. J. Nelson, and M. Ghods, "Technology flexibility: Conceptualization, validation, and measurement," *Proc. Hawaii Int. Conf. Syst. Sci.*, vol. 3, no. February 1997, pp. 76–87, 1997, doi: 10.1109/HICSS.1997.661572.

[11] Y. Arafat, "Fleksibilitas Sistem Informasi dari Perspektif Pengguna Dan Pengembang Sistem Informasi," *Elkha*, vol. 8, no. 1, pp. 37–41, 2016, doi: 10.26418/elkha.v8i1.18226.

[12] J. Gebauer and F. Schober, "Information System Flexibility and the Cost Efficiency of Business Processes," *J. Assoc. Inf. Syst.*, vol. 7, no. 3, pp. 122–147, 2006, doi: 10.17705/1jais.00084.
[13] Y. Gong and M. Janssen, "Measuring process flexibility and agility," *ACM Int. Conf. Proceeding Ser.*, no. June 2014, pp. 173–182, 2010, doi: 10.1145/1930321.1930358.
[14] R. Cognini, F. Corradini, S. Gnesi, A. Polini, and B. Re, "Business process flexibility - a systematic literature review with a software systems perspective," *Inf. Syst. Front.*, vol. 20, no. 2, pp. 343–371, 2018, doi: 10.1007/s10796-016-9678-2.
[15] M. Rosemann and J. Recker, "Context-aware process design: Exploring the extrinsic drivers for process flexibility," *CEUR Workshop Proc.*, vol. 236, pp. 149–158, 2006.
[16] P. Afflerbach, G. Kastner, F. Krause, and M. Röglinger, "An optimization model for valuating process flexibility," *Int. Conf. Inf. Syst. (ICIS 2013) Reshaping Soc. Through Inf. Syst. Des.*, vol. 1, pp. 450–467, 2013.
[17] M. Vakola, "BUSINESS PROCESS RE-ENGINEERING AND ORGANISATIONAL CHANGE: EVALUATION OF IMPLEMENTATION STRATEGIES," *Univ. Salford, Salford, UK*, vol. 1, no. 10, pp. 9–39, 1999.