E-ISSN: 2715-2731

Analisis dan Perancangan Software Simulasi Pertumbuhan Model Proses Bisnis Menggunakan *Random Growth Model*

Fany Parama Admaja[1,*], Mauren Helvia Devi[2], Aditya Prasetyo[3], Muhammad Ainul Yaqin[4]

Jurusan Teknik Informatika, Universitas Islam Negeri Maulana Malik Ibrahim, Indonesia

[1]1admaja404@gmail.com; [2]maurendeviia@gmail.com; [3]adityaprasetyo33@gmail.com; [4]yaqinov@gmail.com
* corresponding author

INFO ARTIKEL

Sejarah Artikel
Diterima: 21 Desember 2020
Direvisi: 1 Februari 2021
Diterbitkan: 30 April 2021

Kata Kunci
Model Proses Bisnis
Simulasi Pertumbuhan
Random Growth Model

ABSTRAK

Tujuan dari penelitian ini adalah untuk menganalisis dan merancang suatu *software* yang berguna untuk melakukan simulasi pertumbuhan model proses bisnis, sehingga didapatkan model proses bisnis yang optimal. Optimal disini memiliki arti bahwa model proses bisnis tersebut tidak lagi bisa dilakukan pertumbuhan. *Input* untuk *software* ini merupakan model proses bisnis. Pada penelitian ini jenis pemodelan yang digunakan adalah *Petri net*. Dalam sekali proses simulasi dibutuhkan dua dua *input*, dimana nilai kompleksitas dari *input* model pertama harus lebih kecil daripada model kedua. Hal ini agar mendapatkan nilai *scalability* yang *scalable*. *Scalability* merupakan potensi yang dimiliki suatu model proses bisnis untuk dapat melakukan pertumbuhan. Semakin besar nilai *scalability* maka semakin memungkinkan untuk terjadi pertumbuhan. Nilai *scalability* inilah yang menjadi kunci utama dari simulasi pertumbuhan tersebut. *Software* akan terus menjalankan proses simulasi dengan syarat nilai *scalability* haruslah ">0" dan "<1". Pertumbuhan dilakukan secara acak dengan menambahkan elemen-elemen baru. Proses simulasi dilakukan dengan dua metode pembobotan, yaitu bercabang dan *sequence*. Hasil dari penelitian ini adalah *software* yang telah dirancang mampu melakukan proses simulasi pertumbuhan dengan menghasilkan nilai *scalability* yang semakin kecil setiap kali pertumbuhan dilakukan. *Software* ini juga menghasilkan *output* berupa file PNML yang mana merupakan hasil dari proses simulasi pertumbuhan tersebut.

PENDAHULUAN

Model proses bisnis merupakan suatu visualisasi terstruktur dari serangkaian kegiatan yang berkaitan dengan suatu aktivitas bisnis [1]. Model proses bisnis sering digunakan dalam perancangan dan analisa suatu sistem. Dalam suatu model proses bisnis terdapat sekumpulan simbol yang memiliki arti berbeda-beda. Terdapat banyak jenis pemodelan yang sering digunakan saat ini, seperti UML, BPMN, EPC, dan Petri Net. Proses bisnis itu sendiri merupakan serangkaian kegiatan yang bekerja sama dalam lingkungan organisasi dan teknis yang bersama-sama mencapai tujuan bisnis [2]. Pada intinya model merupakan gambaran dari suatu sistem atau aktifitas yang tidak bisa kita lihat secara langsung.

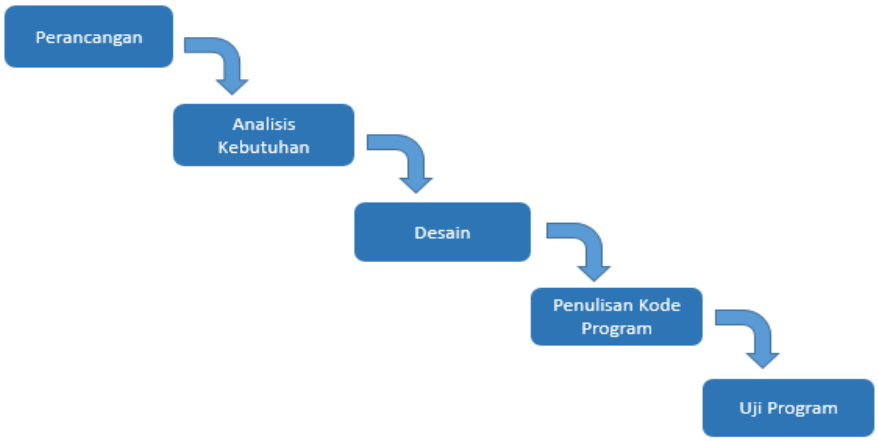
Kesuksesan dari suatu sistem terletak pada proses bisnis yang digunakan. Dalam pengimplementasian proses bisnis dalam kehidupan nyata, setiap badan usaha menggunakan proses yang berbeda beda, meskipun badan usaha tersebut bergerak di bidang yang sama. Namun ada juga yang menerapkan proses bisnis yang sama. Tujuannya tidak lain untuk memaksimalkan keuntungan yang didapat. Karena hal tersebut merupakan indikator keberhasilan dari suatu model proses bisnis yang digunakan. Proses bisnis yang digunakan

pun tidak selamanya tetap. Terkadang ada beberapa faktor yang mengharuskan melakukan perubahan atau penambahan proses bisnis pada suatu sistem yang ada. Oleh karenanya, suatu proses bisnis harus mempunyai fleksibilitas, sehingga proses bisnis dapat divariasikan sesuai dengan kondisi dan kebutuhan. Dalam implementasinya, tidak dapat diketahui apakah proses bisnis yang digunakan sudah optimal atau tidak. Untuk mengetahuinya perlu dilakukan simulasi pertumbuhan model proses bisnis. Simulasi didefinisikan sebagai sekumpulan metode dan aplikasi untuk menirukan atau merepresentasikan perilaku dari sistem nyata [3]. Simulasi memungkinkan seseorang untuk mengimplementasikan model diskrit. Hal ini memungkinkan kita untuk mempertimbangkan keragaman proses produksi. Tujuan simulasi adalah untuk menyeimbangkan kembali elemen-elemen produksi dan memilih parameter yang paling optimal [4]. Proses simulasi dapat dilakukan pada komputer menggunakan *software*. Kunci utama dari simulasi pertumbuhan ini adalah, model proses bisnis harus *scalable*. Hal ini berarti bahwa proses bisnis memiliki kemampuan untuk melakukan pertumbuhan. Untuk melakukan perhitungan terhadap proses bisnis, digunakan rumus *scalability* [5]. *Scalability* didapatkan melalui perhitungan skala dan *similarity*. Rumus-rumus tersebut nantinya diimplementasikan dalam bentuk algoritma sehingga bisa dijalankan pada program di komputer. Pertumbuhan akan dilakukan secara acak dengan pembobotan bercabang dan *sequence*. Pada sistem ini menggunakan model proses bisnis berupa *file* pnml sebagai input dan juga output dari simulasi tersebut. Tujuan dari penelitian ini adalah menghasilkan sebuah *software* yang mampu melakukan perhitungan dan simulasi pertumbuhan model proses bisnis untuk mendapatkan suatu variasi model proses bisnis paling optimal.

Tujuan dari dilakukannya penelitian ini adalah untuk menghasilkan sebuah produk perangkat lunak yang berguna untuk menyimulasikan pertumbuhan model proses bisnis sehingga didapatkan model proses bisnis yang optimal. Hal ini tentu sangat dibutuhkan dalam dunia nyata, salah satunya yaitu dalam pembangunan sebuah *startup.* Karena model proses bisnis merupakan representasi dari aktivitas yang nyata. Oleh karenanya proses bisnis yang harus mampu menunjang dan memaksimalkan operasional. Banyaknya keuntungan yang diperoleh menjadi indikator keberhasilan dari model proses bisnis yang digunakan.

METODE

Dalam perancangan dan analisis *software* simulasi model proses bisnis menggunakan *random growth model* ini, digunakan metode *waterfall* dalam perancangannya. *Waterfall* merupakan metode dan proses pengembangan perangkat lunak tradisional yang sering digunakan dalam proyek-proyek pengerjaan perangkat lunak. Metode *waterfall* menggunakan model sekuensial, sehingga setelah penyelesaian satu set kegiatan, mengakibatkan pada dimulainya aktivitas berikutnya [6]. Alasan penggunaan metode ini karena sesuai dengan objek penelitian yang dilakukan dan juga sering digunakan pada kasus serupa.



Gambar 1. *Waterfall* Model

Pada Gambar 1 merupakan gambaran alur dari *waterfall* model dari penelitian yang akan dilakukan. Pertama terlebih dahulu dilakukan perancangan sistem yang akan dibuat, kemudian berlanjut pada analisis kebutuhan, lalu setelah itu desain, penulisan kode program, dan yang terakhir tahap pengujian.

Perancangan Sistem

Pada tahap ini ditentukan bagaimana cara kerja dan proses dari software yang akan dibuat. Mulai dari pengguna melakukan *input* file PNML, melakukan perhitungan *similarity workflow*, *control flow complexity*, *scalability*, hingga *output* berupa file PNML hasil simulasi pertumbuhan.

Analisis Kebutuhan

Proses pengamatan dan juga pemenuhan kebutuhan suatu sistem dilakukan pada tahap ini. Pertimbangan kebutuhan ini diperlukan agar *software* bisa berjalan dengan lancar. Berbagai aspek yang berpengaruh dalam proses perancangan *software* harus benar-benar diperhatikan dan dipertimbangkan. Diperlukan juga interaksi dengan pengguna untuk mengenai apa saja yang dibutuhkan.

Desain Sistem

Hasil dari analisis kebutuhan yang telah dilakukan sebelumnya, akan dikaji pada tahap ini. Desain yang dibuat berupa desain sistem dan juga desain komponen. Seperti pembuatan interface, pemilihan bahasa pemrograman, kenyamanan pengguna, efisiensi *software*, juga mengenai cara pengelolaan sumber daya yang tersedia.

Penulisan Kode Program

Setelah kita membuat desain software, selanjutnya adalah penulisan kode program. Karena selain faktor-faktor yang telah disebutkan sebelumnya, algoritma merupakan hal yang vital dalam suatu *software*. Karena tanpa adanya algoritma, *software* yang kita buat tidak akan berjalan lancar. Penulisan algoritma haruslah sistematis. Hal ini bertujuan apabila ada kendala atau *error* pada saat tahap pengujian, maka akan lebih mudah dilakukan perbaikan. Algoritma yang dibuat harus sesuai dengan rumus perhitungan dalam proses simulasi. Seperti rumus perhitungan *similarity workflow*, *control flow complexity*, skala model, dan *scalability*.

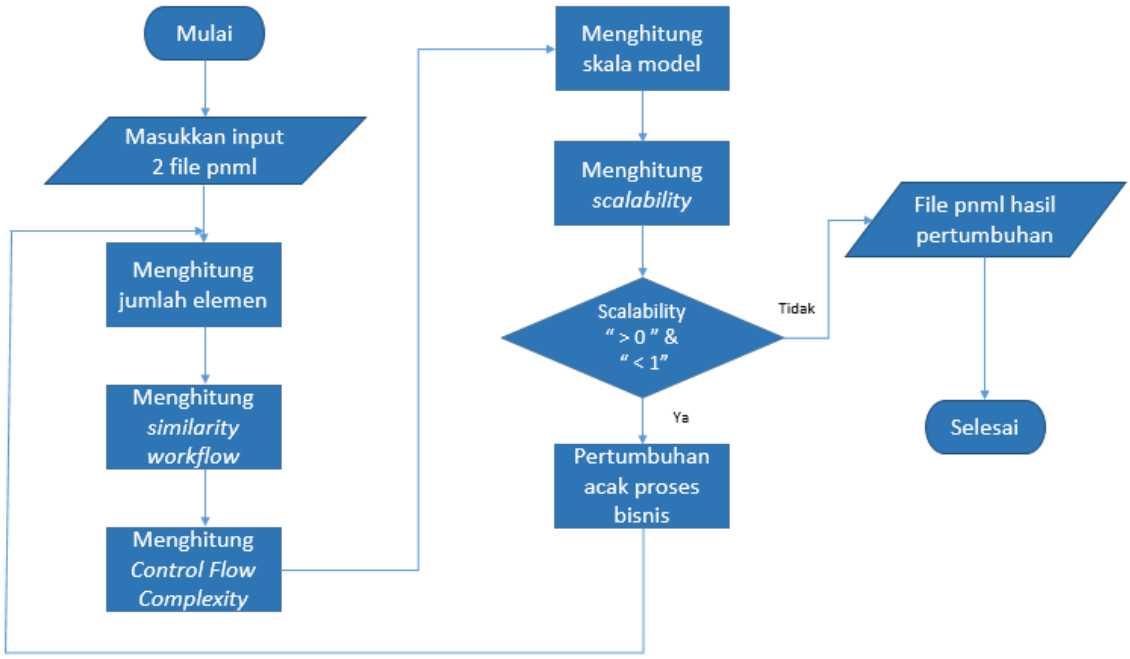
Uji Program

Tahap ini merupakan tahap akhir dari serangkaian proses yang telah dilakukan sebelumnya. Tujuannya adalah untuk mengetahui hasil dari *software* yang telah dibuat, apakah sesuai dengan yang diharapkan atau tidak. Tahap ini juga digunakan untuk mencari kesalahan pada tahap sebelumnya.

HASIL DAN PEMBAHASAN

Alur Program

Setiap *software* pasti memiliki *flowchart*. *Flowchart* ini berguna agar mudah memperoleh informasi dan melakukan pengamatan mengenai bagaimana jalannya suatu program. Mulai dari memasukkan inputan, hingga mengolahnya menjadi output yang diinginkan. Sama halnya dengan software simulasi pertumbuhan model proses bisnis dengan menggunakan *random growth model* ini. Untuk alur kerjanya bisa dilihat pada Gambar 2.



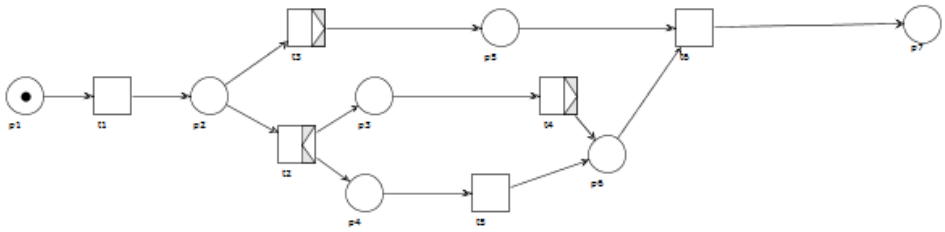
Gambar 2. Flowchart Software Simulasi Pertumbuhan Model Proses Bisnis Dengan Random Growth Model

Pada Gambar 2, berbagai simbol yang saling terhubung satu sama lain dengan tanda panah. *flowchart* dimulai dengan simbol terminator yang mana merupakan awal dimulainya program. Lalu diikuti objek berbentuk jajar genjang yang melambangkan input. Input yang dimasukkan merupakan model proses bisnis berupa *file Petri Net*. Setelah itu terdapat simbol berbentuk persegi panjang yang melambangkan suatu proses yang dijalankan oleh *software*. Proses pertama yang dilakukan adalah melakukan perhitungan jumlah elemen dari kedua model proses bisnis yang diinputkan sebelumnya. Jumlah elemen kedua model tersebut harus berbeda. Dimana jumlah elemen model pertama harus lebih sedikit dari model kedua. Jumlah elemen yang didapat dari kedua model tersebut akan digunakan untuk menghitung nilai similarity, baik secara s*tructural* maupun *behavioral*. Setelah itu program akan menghitung *Control Flow Complexity*, yang mana hasil tersebut digunakan untuk menghitung besar skala model pada perhitungan selanjutnya. Kemudian dilakukan perhitungan nilai *scalability*, yang mana merupakan parameter dari simulasi ini. Setelah nilai *scalability* didapatkan, terdapat objek berbentuk belah ketupat yang melambangkan

decision. Pada tahap inilah proses simulasi pertumbuhan akan terjadi. Namun terdapat kondisi yang harus terpenuhi agar pertumbuhan bisa terjadi. Dimana, nilai *scalability* harus ">0" dan "<1". Apabila syarat terpenuhi maka proses simulasi akan berlangsung. Proses itu akan terus berulang hingga akhirnya kondisi tidak lagi terpenuhi, dan simulasi berhenti. *Output* yang dihasilkan dari simulasi pertumbuhan proses bisnis ini adalah file PNML baru yang sudah dilakukan pertumbuhan sehingga lebih optimal daripada sebelumnya. Suatu model proses bisnis dikatakan optimal apabila sudah tidak *scalable*, atau tidak memungkinkan untuk dilakukan pertumbuhan lagi.

Input Model Proses Bisnis

Untuk memulai proses simulasi, dibutuhkan *input* dua model proses bisnis berupa file PNML. Model proses bisnis yang diinputkan memiliki syarat yang harus dipenuhi, dimana jumlah elemen dari model pertama harus lebih kecil dari model kedua. Hal ini perlu diperhatikan agar mendapatkan nilai *scalability* yang *scalable*. Salah satu model proses bisnis menggunakan Petri Net dapat dilihat pada Gambar 3.



Gambar 3. Contoh Model Proses Bisnis Menggunakan Petri Net

Perhitungan Jumlah Elemen

Setelah melakukan *input* menggunakan dua model proses bisnis, tahap selanjutnya program akan melakukan perhitungan jumlah elemen dari kedua model proses bisnis. Caranya yaitu dengan perlakukan *parsing* terhadap atribut didalam file PNML tadi, karena pada dasarnya file PNML berbasis XML. Caranya yaitu dengan memanggil nama tag yang diinginkan seperti *place, transition*, dan *arc*. Elemen-elemen tersebut kemudian disimpan ke dalam *ArrayList*. *ArrayList* dipilih karena bersifat fleksibel, tanpa perlu menentukan ukuran di awal. Jumlah elemen dari model proses bisnis diambil berdasarkan ukuran *ArrayList* yang telah terisi.

Perhitungan *Similarity Workflow*

Proses selanjutnya dari *software* ini adalah mencari nilai *similarity*, baik secara *structural* maupun *behavioral*. Perhitungan kesamaan dengan structural similarity, lebih mengarah pada struktur dan jumlah elemen yang ada didalam suatu model proses bisnis. Sedangkan behavioral similarity lebih mengarah kepada kesamaan relasi antar aktivitas pada model proses bisnis [7]. *Software* ini menggunakan algoritma *Jaccard Coefficient*. Untuk rumusnya dapat dilihat pada Persamaan 1.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \quad (1)$$

Dimana : J (A,B) = Nilai Jaccard dari model proses bisnis A dan B.

$A \cap B$ = Intersect dari model proses bisnis A dan B.

$A \cup B$ = Gabungan dari model proses bisnis A dan B.

Perhitungan Control Flow Complexity

Perhitungan *Control Flow Complexity* dilakukan pada setiap model proses bisnis yang telah diinputkan. Tujuannya yaitu untuk mencari tingkat kompleksitas dari suatu model proses bisnis. Caranya yaitu dengan menghitung jumlah logika AND-*split* dan XOR-*split* pada model proses bisnis. Apabila suatu model proses bisnis tidak memiliki percabangan didalamnya atau bisa dikatakan *sequence*, maka nilai *Control Flow Complexity* dipastikan memiliki nilai satu. Untuk mengetahui rumusnya, bisa dilihat pada Persamaan 2.

$$CFC\ (x) = \sum CFC_{XOR-split}\ (x) + \sum CFC_{OR-split}\ (x) + \sum CFC_{AND-split}\ (x) \quad (2)$$

Dimana : $CFC_{XOR\text{-}split}(x)$ = jumlah percabangan (x)
$CFC_{OR\text{-}split}(x) = 2^{(\text{jumlah percabangan(x) - 1})}$
$CFC_{AND\text{-}split}(x) = 1$

Perhitungan Skala Model

Selain nilai *similarity*, nilai skala model juga merupakan faktor penting untuk memperoleh nilai *scalability*. Untuk mendapatkanya cukup mudah, yaitu tinggal mengalikan jumlah elemen dengan nilai *Control Flow Complexity* yang telah didapatkan pada perhitungan tahap sebelumnya. Untuk rumus menghitung skala bisa dilihat pada persamaan 3.

$$Skala\ (A) = E(A) * CFC\ (A) \quad (3)$$

Perhitungan Scalability

Nilai *scalability* merupakan nilai penting yang menjadi parameter dalam proses simulasi pertumbuhan model proses bisnis ini. Jadi berapapun nilai yang dihasilkan akan berpengaruh terhadap simulasi itu sendiri. Untuk rumus mencari scalability bisa dilihat pada persamaan 4.

$$Scalability\ (A,B) = 1 - (Skala\ (A,B) * Sim\ (A,B)) \quad (4)$$

Pada Persamaan 4 , nilai similarity yang digunakan merupakan rata-rata antara *structural similarity* dan *behavioral similarity*. Sedangkan nilai skala didapatkan melalui perbandingan skala model proses bisnis A, dan B.

Pertumbuhan Acak Proses Bisnis

Proses pertumbuhan proses bisnis dilakukan dengan memperhatikan kondisi yang harus terpenuhi. Pertumbuhan dapat terjadi apabila nilai *scalability* “>0” dan “<1”. Proses pertumbuhan akan terus berlangsung selama kondisi terpenuhi. Itulah mengapa pada hasil *scalability* pada iterasi pertama haruslah *scalable*. Untuk mendapatkan nilai scalability yang *scalable*, maka model proses bisnis A harus memiliki jumlah elemen lebih sedikit daripada model B. Pertumbuhan elemen-elemen baru pada model proses bisnis dilakukan secara acak. Algoritma pengacakan yang digunakan pada penelitian ini adalah *Linear Congruential Generator* (LCG). Rumus dari LCG bisa dilihat dalam Persamaan 5.

$$X_{n+1} = \left((a\ X_n) + c\ \right) mod\ m \quad (5)$$

Dimana : Xn = bilangan acak ke-n
Xn + 1 = bilangan acak pada iterasi selanjutnya
a = faktor pengali
c = pertambahan nilai

m = modulus

Pada Persamaan 5, kunci dari pengacakan menggunakan algoritma LCG ini adalah X_0 atau disebut dengan nilai umpan. Nilai ini memiliki pengaruh terhadap bilangan acak yang dihasilkan pada iterasi selanjutnya. Penentuan konstanta a, c dan m juga memiliki pengaruh terhadap kualitas bilangan acak yang dibangkitkan dalam arti bilangan acak yang didapatkan benar-benar merupakan hasil pengulangan. Penentuan nilai m akan menjadi batas maksimal nilai yang dihasilkan. Artinya, hasil dari pengacakan tersebut tidak akan memiliki nilai lebih besar dari nilai m [8].

Hasil Uji Coba Program

Pada uji coba ini, akan dilakukan simulasi pertumbuhan model proses bisnis dengan menggunakan 3 file PNML**.** Model-model tersebut memiliki struktur dan jumlah elemen yang berbeda, dimana model A< model B< model C. Untuk lebih detailnya, bisa melihat pada Gambar 4.

Model A : 17 Elemen
- Place : p1,p2,p3,p4,p5
- Transition : t4,t1,t2,t3
- Arc : p1 -> t1, t1 -> p2, p2 -> t2, t2 -> p3, p3 -> t3, t3 -> p4, p4 -> t4, t4 -> p5

Model B : 22 Elemen
- Place : p1,p2,p3,p4,p5,p6
- Transition : t4,t5,t1,t2_op_1,t3
- Arc : p1 -> t1, t4 -> p5, t1 -> p2, p3 -> t4, t5 -> p6, p5 -> t5, p2 -> t3, t3 -> p3, p4 -> t4, p2 -> t2_op_1, t2_op_1 -> p4

Model C : 27 Elemen
- Place : p1,p2,p3,p4,p5,p6,p7
- Transition : t4_op_1,t5,t6,t1,t2_op_1,t3_op_1
- Arc : t6 -> p7, p4 -> t5, p6 -> t6, t5 -> p6, p1 -> t1, t1 -> p2, p5 -> t6, t3_op_1 -> p5, p2 -> t3_op_1, t2_op_1 -> p4, p2 -> t2_op_1, t2_op_1 -> p3, p3 -> t4_op_1, t4_op_1 -> p6

Gambar 4. Elemen- Elemen Pada Model Proses Bisnis A,B, dan C

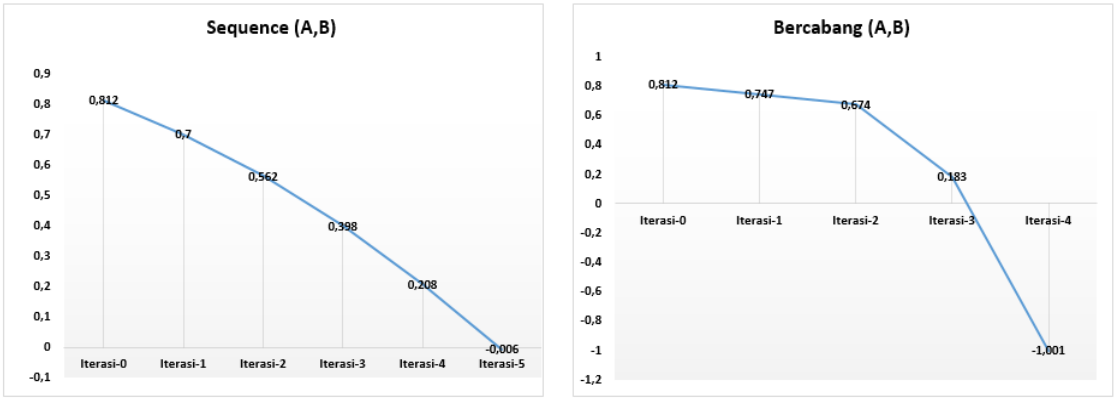
Pada Gambar 4 dapat diketahui jumlah elemen model proses bisnis A lebih sedikit dari model B. Begitupun jumlah elemen pada model B lebih sedikit dari model C. Hal ini agar bertujuan mendapatkan nilai *scalability* yang *scalable.* Uji coba akan dilakukan sebanyak tiga kali, yaitu membandingkan model A dengan B, A dengan C, lalu kemudian B dengan C. Setelah itu program melakukan perhitungan *similarity*, skala, *Control Flow Complexity* hingga didapatkan variasi nilai yang bisa dilihat pada Tabel 1.

Tabel 1. Hasil Perbandingan Nilai Scalability

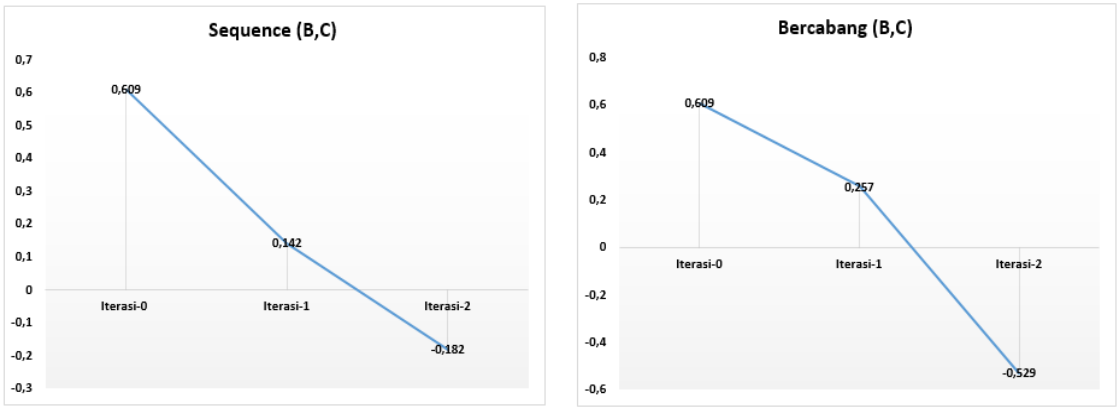
	A	B	C
A	0	-	-
B	0.812	0	-
C	0.900	0.609	0

Dari Tabel 1, diperoleh beragam nilai *scalability* hasil perbandingan antara model A, B,dan C. Dapat diketahui apabila jumlah elemen pada input pertama dan input kedua sama, maka *scalability* akan menghasilkan nilai "0". Sedangkan apabila jumlah elemen pada *input* pertama lebih besar daripada *input* kedua, maka tabel diisi dengan tanda "-", yang artinya nilai yang dihasilkan sudah pasti tidak dapat dilakukan pertumbuhan, atau tidak *scalable*.

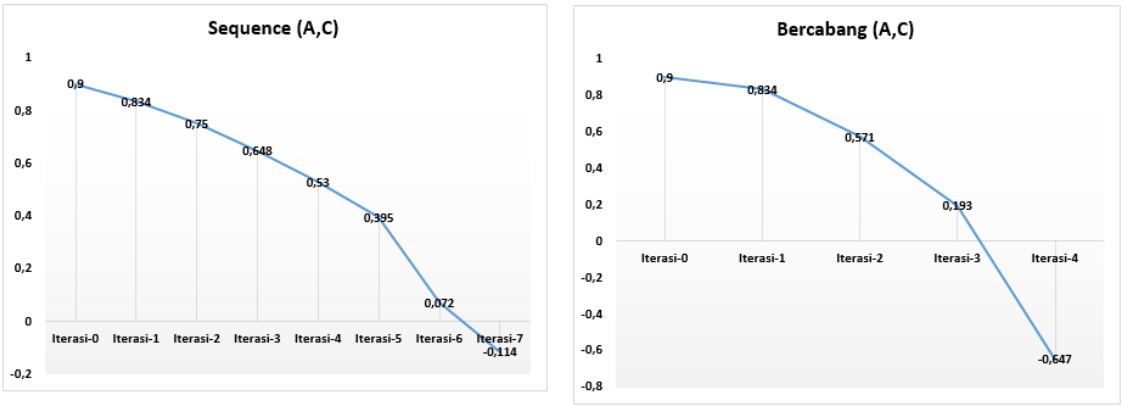
Simulasi pertumbuhan dilakukan dengan dua macam pembobotan, yaitu bercabang dan sequence. Dari kedua pembobotan tersebut nantinya akan didapatkan nilai *scalability* yang berbeda pada setiap iterasinya. Bukan hanya itu, jumlah banyaknya iterasi pada hasil simulasi juga berbeda beda. Namun tidak menutup kemungkinan iterasi yang dihasilkan memiliki jumlah yang sama. Proses simulasi akan terus terjadi selama kondisi yang telah ditentukan terpenuhi, yaitu scalability harus bernilai ">0" dan "<1". Hasil simulasi pertumbuhan model proses bisnis ditunjukkan pada Gambar 5, Gambar 6, dan Gambar 7. Bisa dilihat bahwa semakin bertambah iterasi nilai *scalability* juga mengalami penurunan. Hal itu menunjukkan bahwa jumlah elemen dan kompleksitas pada model proses bisnis semakin bertambah. Iterasi berhenti tepat setelah *scalability* bernilai minus. Itu artinya model proses bisnis sudah tidak memenuhi kondisi yang ditentukan untuk bisa melakukan pertumbuhan lagi. Sehingga proses bisnis dan nilai scalability paling optimal yaitu tepat sebelum iterasi terakhir. Untuk contoh output hasil simulasi bisa dilihat pada Gambar 8.



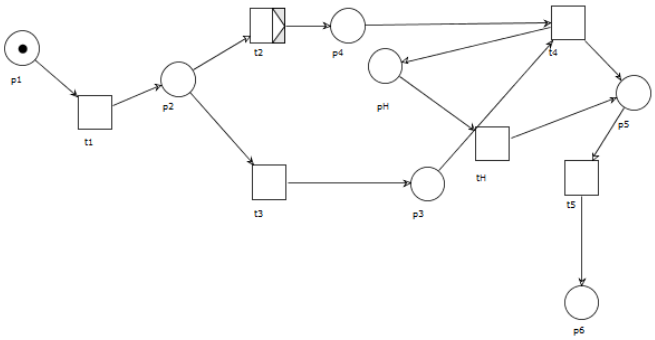
Gambar 5. Hasil Simulasi Pertumbuhan Model Proses Bisnis A dan B



Gambar 6. Hasil Simulasi Pertumbuhan Model Proses Bisnis B dan C



Gambar 7. Hasil Simulasi Pertumbuhan Model Proses Bisnis A dan C



Gambar 8. Model Proses Bisnis Hasil Simulasi Pertumbuhan

KESIMPULAN

Berdasarkan penelitian yang telah dilakukan, dapat disimpulkan bahwa proses perhitungan *similarity*, *Control Flow Complexity*, *scalability* dapat diimplementasikan menjadi algoritma di dalam sebuah *software*. Hal ini tentunya membuat proses simulasi pertumbuhan model proses bisnis menjadi lebih cepat dan mudah. Penerapan *random growth model* pada *software* menggunakan algoritma *linear congruential generator* terbilang sukses. Indikator yang dapat kita lihat yaitu hasil dari simulasi memiliki nilai yang bervariasi. Jumlah elemen, *similarity workflow*, *Control Flow Complexity*, dan skala model merupakan hal yang berpengaruh apabila ingin mendapatkan nilai *scalability* yang *scalable.* Semakin besar nilai scalability maka semakin memungkinkan untuk dilakukan pertumbuhan. Besar kecilnya nilai s*calability* pada model proses bisnis, tidak mempengaruhi banyaknya iterasi pada proses simulasi. Hal ini diakibatkan oleh pengimplementasian *random growth model*. Penelitian terbukti berhasil, ditunjukkan dengan nilai *scalability* yang mengalami penurunan pada setiap iterasinya.

REFERENSI

[1] A. P. Wahyu, M. A. Yaqin, and S. Zaman, "Common Process Extraction Pada Scalable Model Proses Bisnis," *Konf. Nas. Sist. Inf.*, no. May, pp. 8–9, 2018.

[2] M. A. Yaqin, R. Adawiyah, W. Ningrum, and A. Janan, *Simulasi Model Proses Bisnis pada Permainan Travel Agency*. 2019.

[3] M. A. Yaqin, E. F. Febriana, Y. Rahmawati, and N. R. P, "Simulasi Model Proses Bisnis pada Permainan Hay Day," *Semin. Nas. Inov. dan Apl. Teknol. di Ind. 2019*, no. February, pp. 20–29, 2019.

[4] I. V. Revina and E. N. Trifonova, "Simulation modeling of the assembly process," *J. Phys. Conf. Ser.*,

vol. 1441, no. 1, 2020, doi: 10.1088/1742-6596/1441/1/012110.

[5] S. Muslihaeny, M. A. Yaqin, and S. Zaman, "Simulasi Pertumbuhan Scalable Business Process Model pada ERP Pondok Pesantren berbasis Production Rule Cellular Automata," vol. 1, no. 2, pp. 30–38, 2019.

[6] I. Fahrurrozi and A. SN, "Proses pemodelan software dengan metode waterfall dan extreme programming : Studi kasus perbandingan," *Univ. Gajah Mada*, pp. 1–10, 2015, [Online]. Available: mediakom-penerbit.com.

[7] M. A. Yaqin, R. Sarno, and A. C. Fauzan, "Scalability measurement of business process model using business processes similarity and complexity," *Int. Conf. Electr. Eng. Comput. Sci. Informatics*, vol. 2017-Decem, no. September, 2017, doi: 10.1109/EECSI.2017.8239129.

[8] B. Prasetyo, I. Agustina, and M. Gufroni, "Perancangan Game Puzzle Pemadam Kebakaran Menggunakan Metode Linear Congruential Generator (LCG)," *JOINTECS (Journal Inf. Technol. Comput. Sci.*, vol. 2, no. 2, pp. 67–72, 2017, doi: 10.31328/jointecs.v2i2.473.