SIMULASI PERTUMBUHAN *SCALABLE BUSINESS PROCESS MODEL* PADA ERP PONDOK PESANTREN BERBASIS *PRODUCTION RULE CELLULAR AUTOMATA*

SKRIPSI

Oleh :
SITI MUSLIHAENY
NIM. 13650054

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2018

SIMULASI PERTUMBUHAN *SCALABLE BUSINESS PROCESS MODEL* PADA ERP PONDOK PESANTREN BERBASIS *PRODUCTION RULE CELLULAR AUTOMATA*

SKRIPSI

Diajukan kepada:
Universitas Islam Negeri Maulana Malik Ibrahim Malang
Untuk memenuhi Salah Satu Persyaratan dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)
Oleh :

SITI MUSLIHAENY
NIM. 13650054

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2018

LEMBAR PERSETUJUAN

SIMULASI PERTUMBUHAN *SCALABLE BUSINESS PROCESS MODEL* PADA ERP PONDOK PESANTREN BERBASIS *PRODUCTION RULE CELLULAR AUTOMATA*

SKRIPSI

Oleh :

SITI MUSLIHAENY
NIM. 13650054

Telah Diperiksa dan Disetujui untuk Diuji
Tanggal: 27 April 2018

Pembimbing I	Pembimbing II
M. Ainul Yaqin, M.Kom	Syahiduz Zaman, M.Kom
NIP.19761013 200604 1 004	NIP. 19700502 200501 1 005

Mengetahui,
Ketua Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Cahyo Crysdian
NIP. 19740424 200901 1 008

LEMBAR PENGESAHAN

SIMULASI PERTUMBUHAN *SCALABLE BUSINESS PROCESS MODEL* PADA ERP PONDOK PESANTREN BERBASIS *PRODUCTION RULE CELLULAR AUTOMATA*

SKRIPSI

Oleh :

SITI MUSLIHAENY
NIM. 13650054

Telah Dipertahankan di Depan Dewan Penguji Skripsi
dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer (S.Kom)
Tanggal: 4 Juni 2018

Susunan Dewan Penguji			**Tanda Tangan**
Penguji Utama	:	Dr. Suhartono, M.Kom NIP. 19680519 200312 1 001	( )
Ketua Penguji	:	M. Fatchurrohman, M.Kom NIP. 19700731 200501 1 002	( )
Sekretaris Penguji	:	M. Ainul Yaqin, M.Kom NIP. 19761013 200604 1 004	( )
Anggota Penguji	:	Syahiduz Zaman, M.Kom NIP. 19700502 200501 1 005	( )

Mengesahkan,
Ketua Jurusan Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Cahyo Crysdian
NIP. 19740424 200901 1 008

PERNYATAAN ORISINALITAS TULISAN

Saya yang bertanda tangan di bawah ini:

Nama : SITI MUSLIHAENY

NIM : 13650054

Jurusan : Teknik Informatika

Fakultas : Sains dan Teknologi

Judul Penelitian : "**SIMULASI PERTUMBUHAN *SCALABLE BUSINESS PROCESS MODEL* PADA ERP PONDOK PESANTREN BERBASIS *PRODUCTION RULE CELLULAR AUTOMATA***"

Menyatakan dengan sebenarnya bahwa skripsi yang saya tulis benar-benar merupakan hasil karya sendiri, bukan merupakan pengambil alihan data, tulisan atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka. Apabila dikemudian hari terbukti atau dapat dibuktikan ini hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 20 April 2018

ng membuat pernyataan

Siti Muslihaeny

NIM. 13650054

HALAMAN PERSEMBAHAN

بِسْمِ اللهِ الرَّحْمٰنِ الرَّحِيمِ

Alhamdulillah puji syukur kehadirat Allah SWT yang telah memberikan kekuatan kepada saya sehingga dapat menyelesaikan studi S1 di kampus tercinta UIN Maliki Malang ini. Sholawat serta salam kepada Nabi Muhammad SAW, yang membawa petunjuk kepada seluruh umat manusia.

Terima kasih kepada Ibu dan Ayah saya yang telah mendidik dan merawat dari kecil hingga sekarang. Senantiasa memberikan motivasi, dorongan spiritual, dan yang setiap hari tanpa lelah mendoakan untuk keberhasilan saya. Mendukung dalam keputusan yang saya pilih dan memberikan kekuatan mental dalam mengerjakan skripsi sehingga saya mampu menyelesaikan segala kewajiban di bangku pendidikan. Terima kasih juga kepada kakak saya tercinta, Riana Susanti yang selalu memberikan semangat dan nasihat untuk saya, dan kepada Adik saya tersayang, Aulia Kholifatuz Zahra yang selalu dapat membuat saya tersenyum dengan tingkah lucunya sehingga mampu memberikan hiburan tersendiri dan membuat pengerjaan skripsi lebih menyenangkan. Terima kasih kepada seluruh keluarga, sahabat yang telah memberikan dukungan penuh kepada saya.

Terima kasih kepada Sahabatku Nunung Nur Laila Varychatin, Yaya dan Ulfy Yulia MH yang selalu memberikan semangat dan berbagai komentar-komentar yang dapat memotivasi pengerjaan skripsi ini. Kepada sahabatku Novita Pratiwi, Linda Mutiara Chanastalia, Lailia Khoirunnisa', Ning Navisa dan Nadiya Fikriyatuz Zakiyah yang selalu sigap membantu dan memberikan dukungan satu sama lain dalam

menyelesaikan skripsi ini. Kepada teman-teman seperjuangan: Awwalia Nur Hayati, Dian Fitriani, Permata Rahmatul Hijah, Dwi Rahayu Utami, Elfiyatul Fitriyah, Risti Nurarifah dan Lin Farihah yang selalu memberikan semangat dan motivasi satu sama lain dalam pengerjaan skripsi ini.

Karya ini saya persembahkan kepada keluarga serta sahabat yang telah mendukung saya selama ini.

MOTTO

"Untuk mendapatkan sesuatu yang kau inginkan, kau harus bersabar dengan sesuatu yang kau benci"

(Imam Ghazali)

"Dream as if you'll live forever, live as if you'll die together."

"Bermimpilah seakan kau akan hidup selamanya, Hiduplah seakan kau akan mati hari ini."

(J.K. Rowling)

KATA PENGANTAR

Assalamu'alaikum Wr.Wb.

Segala puji syukur kehadirat Allah SWT tuhan semesta alam, karena atas segala rahmat dan karunia-Nya sehingga penulis mampu menyelesaikan skripsi yang berjudul "Simulasi Pertumbuhan *Scalable Business Process Model* pada ERP Pondok Pesantren Berbasis *Production Rule Cellular Automata*" dengan baik dan dapat menyelesaikan studi di Universitas Islam Negeri Maulana Malik Ibrahim Malang. Sholawat serta salam selalu tercurah kepada tauladan terbaik Nabi Muhammad SAW yang membawa petunjuk kepada seluruh umat manusia dari zaman kebodohan menuju islam yang *rahmatan lil alamiin.*

Dalam menyelesaikan skripsi ini, banyak pihak yang telah memberikan bantuan baik secara moril, materiil, nasihat dan semangat. Atas segala bantuan yang telah diberikan, penulis ingin menyampaikan doa dan ucapan terima kasih yang sedalam-dalamnya kepada:

1. Bapak Prof. Dr. Abdul Haris, M.Ag selaku rektor Universitas Islam Negeri Maulana Malik Ibrahim Malang.
2. Ibu Dr. Sri Harini, M.Si, selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
3. Bapak Dr. Cahyo Crysdian, selaku ketua jurusan Teknik Informatika yang telah memberikan motivasi untuk terus berjuang.
4. Bapak M. Ainul Yaqin, M.Kom, selaku dosen pembimbing I dan Bapak H. Syahiduz Zaman, M.Kom, selaku dosen pembimbing II yang telah meluangkan waktu untuk membimbing, mengarahkan dan memberi

masukan dan nasihat kepada penulis dalam pengerjaan skripsi ini hingga akhir.

5. Bapak Dr. Mokhamad Amin Haryadi, MT, selaku Dosen Wali yang memberikan motivasi dan saran untuk kebaikan penulis.
6. Segenap dosen teknik informatika yang telah memberikan bimbingan keilmuan kepada penulis selama masa studi.
7. Kedua orang tua serta keluargaku yang telah memberikan semangat, motivasi, dukungan dan kasih sayang melimpah sehingga terselesaikan studi S1 di kampus ini.
8. Teman-teman seperjuangan teknik informatika FBI-G, FBI, Skripsi Sukses FORTINITY 2013 yang telah memberikan semangat, motivasi, dukungan dan bantuan yang sangat berharga dalam menyelesaikan skripsi ini.
9. Semua pihak yang ikut dalam berkontribusi dalam membantu menyelesaikan skripsi ini.

Berbagai kekurangan dan kesalahan mungkin pembaca temukan dalam penulisan skripsi ini, untuk itu penulis menerima segala kritik dan saran yang membangun dari pembaca sekalian. Semoga apa yang menjadi kekurangan bisa disempurnakan oleh peneliti selanjutnya dan semoga karya ini senantiasa dapat memberikan manfaat. Amim. *Wassalamualaikum Wr.Wb*

Malang, 16 April 2018

Siti Muslihaeny

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE SUMBER

ABSTRAK

Muslihaeny, Siti. 2018. **Simulasi Pertumbuhan *Scalable Business Process Model* pada ERP Pondok Pesantren Berbasis *Production Rule Cellular Automata*.** Skripsi. Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
Pembimbing (I) Muhammad Ainul Yaqin, M.Kom (II) Syahiduz Zaman, M.Kom

Kata Kunci: Proses Bisnis Pondok Pesantren, Model *Petri Net, Control Flow Complexity, Scalability,* Simulasi Pertumbuhan Proses Bisnis, *Production Rule Cellular Automata.*

Peningkatan kapasitas produksi dalam *enterprise* dapat menjadi pemicu meningkatnya proses produksi dan proses bisnis. Peningkatan kapasistas produksi dan proses bisnis dapat diprediksi menggunakan simulasi pertumbuhan proses bisnis. Namun, simulasi pertumbuhan proses bisnis harus disesuaikan kembali dengan kebutuhan *enterprise*. Contoh kebutuhan *enterprise* yang menjadi pemicu tumbuhnya proses bisnis pada studi kasus Pondok Pesantren yaitu pertambahan jumlah santri dan kurikulum, sehingga memicu adanya penambahan kelas, penambahan jam mata pelajaran dan lain sebagainya.

Penelitian ini bertujuan untuk mensimulasi *scalable business process model* dari ERP Pondok Pesantren guna mendapatkan variasi proses bisnis yang mungkin terjadi menggunakan teori *Production Rule Cellular Automata*. Inputan untuk sistem ini adalah model proses bisnis dari empat tipe Pondok Pesantren yang dimodelkan menggunakan *Petri net* berupa *file* PNML. Parameter yang digunakan untuk mensimulasikan pertumbuhan proses bisnis adalah *scalability*. *Scalability* / skalabilitas adalah potensi proses bisnis untuk tumbuh atau kemampuan pertumbuhan dari proses bisnis. Nilai *scalability* dapat diukur dengan perhitungan kemiripan *workflow* dan skala model proses bisnis. Pertumbuhan proses bisnis terjadi pada model A dengan pembanding model B. Syarat untuk dapat terjadi pertumbuhan pada proses bisnis yaitu *complexity* model A harus lebih kecil daripada model B. Pola pertumbuhan proses bisnis dilakukan secara *random* dengan dua pembobotan yaitu secara percabangan dan *sequence*. Pertumbuhan berhenti jika *scalabiltiy* pada nilai ">=0" dan "<1" Hasil dari penelitian ini menunjukkan bahwa sistem mampu melakukan simulasi pertumbuhan pada *file* PNML yang ditandai dengan nilai *recent scalability* lebih kecil dibandingkan *scalability* awal. Kemudian *output* sistem adalah *file* PNML hasil dari simulasi pertumbuhan proses bisnis dengan pertumbuhan elemen baru.

Berdasarkan hasil pengujian dari segi nilai *scalability*, penelitian terbukti berhasil dengan fakta penurunan nilai *scalability*.

ABSTRACT

Muslihaeny, Siti. 2018. **The Growth Simulation of Scalable Business Process Model in ERP of Islamic Boarding School with Production Rule Cellular Automata Based.** Undergraduate Thesis. Informatics Engineering Department, Faculty of Science and Technology, State Islamic University of Maulana Malik Ibrahim Malang

Advisor (I) Muhammad Ainul Yaqin, M.Kom (II) Syahiduz Zaman, M.Kom

Keyword: Business Process of Islamic Boarding School, Petri Net Model, Control Flow Complexity, Scalability, the Growth Simulation of Business Process, Production Rule Cellular Automata.

The increase of production capacity in the enterprise can be a trigger for increased production and business processes. The increase of production and business processes can be predicted using a growth simulation of the business process. However, the growth simulation of the business processes must be adjusted to the enterprise needs. The examples of enterprise needs that trigger the growth of business process in boarding schools' case study are the increase of the number of students and the curriculum, so that creates the addition classes, increased hours of subjects and others.

The aim of this research is to simulate a scalable business process model of ERP boarding schools in order to get business process variations that may occur using the theory of Cellular Automata Rule Productions. The input for this system is the model the business processes of the four types of boarding schools which are modeled using Petri net in the form of PNML files. The parameter that is used to simulate the growth of business processes is scalability. Scalability/business process is a potential scalability to grow or the ability of the growth of the business processes. The value of scalability can be measured by the calculation of similarity scale models and workflow business process. The growth of business process occurs in comparison with model A model B. A condition for growth can occur in a business process that is A model of complexity should be smaller than the model B. The pattern of the business growth processes is carried out with two random weighting such as branching and sequencing. The growth in the value of scalability stop if "> = 0" and "1" the result of < research indicates that the system is able to perform growth simulation on PNML files marked with a value smaller than recent early scalability. Then, the output of the system is the result of simulation PNML file growth business processes with the growth of the new element.

Based on the test, results in terms of the value of scalability, the research proved successful with the decline in the value of scalability.

ملخص البحث

مصلحيني ، سيتي. 2018. **محاكاة النمو لنموذج عملية قابلة للتطوير (*Scalable Business Process Model*) في تخطيط موارد المشاريع (ERP) المؤسسات الاسلامية القائمة على أوتوماتا للخلوية القاعدة الإنتاج (*Production Rule Cellular Automata*)**. البحث الجامعي. قسم المعلوماتية كلية العلوم والتكنولوجيا الجامعة الإسلامية الحكومية مولانا مالك إبراهيم مالانج.

الاشراف: محمد عين اليقين، الماجستير، وشهيد الزمان، الماجستير

الكلمات الرئيسية: العمليات التجارية للمؤسسات الاسلامية، نموذج *Petri Net*، تعقيد تدفق التحكم (*Control Flow Complexity*)، التدرجية، محاكاة نمو العمليات التجارية ، أوتوماتا للخلوية القاعدة الإنتاج

زيادة الطاقة الإنتاجية فى زيادة الاعمال تمكن أن تزيد عمليات الإنتاج والعمليات التجارية. زيادة القدرة الإنتاجية والعمليات التجارية هي باستخدام عمليات محاكاة نمو العمليات التجارية. ومع ذلك، تجب إعادة محاكاة نمو لعملية الأعمال مع احتياجات زيادة الاعمال. أمثلة احتياجات الشركات التي تؤدي إلى نمو العمليات التجارية في دراسة حالة المؤسسات الاسلاميةهي زيادة عدد الطلاب والمنهج الدراسية، مما يؤدي إلى إضافة الفصل، إضافة ساعات من الموضوعات وغيرها

يهدف هذا البحث لان يحك نموذج عملية قابلة للتطوير من تخطيط موارد المؤسسات الاسلامية للحصول على اختلافات في العمليات التجارية التي قد تحدث باستخدام نظرية أوتوماتا للخلوية القاعدة الإنتاج. مدخلات لهذا النظام هو نموذج عملية الأعمال من أربعة أنواع من المؤسسات الاسلامية من *Petri Net* في شكل ملفات PNML المعلمات المستخدمة لمحاكاة نمو العمليات التجارية هي التدرجية. التدرجية هي إمكانية نمو عمليات الأعمال أو قدرات نمو عملية الأعمال. تقيس قيمة التدرجية عن طريق حساب تشابه مسار العمل *workflow* ومقياس نموذج عملية الأعمال. يحدث نمو عملية الأعمال في النموذج (أ) مع نموذج المقارنة (ب). الشرط النمو المطلوب في عملية الأعمال هو نموذج التعقيد أ فهو أصغر من النموذج ب. نمط نمو عملية الأعمال هو عشوائي مع اثنين من الترجيح اي المتفرعة والتسلسل. يتوقف النمو إذا كان التدرجية فى ”0=<“ و ”1>“. دلت نتائج هذا البحث إلى أن النظام هو قادر على محاكاة النمو في ملفات PNML التي تتميز

بقيم التدرجية الحديثة الأصغر من التدرجية الأولي. إن إخراج النظام هو ملف PNML النتيجة من محاكاة نمو عمليات الأعمال مع نمو عناصر جديدة.

واستنادًا إلى نتائج الاختبار من قيمة التدرجية، أثبتت البحث نجاحا مع حقيقة الانخفاض للقيمة التدرجية

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Kinerja proses bisnis merupakan elemen penting dari sebuah *enterprise*. Semakin baik proses bisnis maka semakin baik pula *enterprise* tersebut (Essam, et al., 2011). Kebutuhan *enterprise* pada saat proses produksi membutuhkan sebuah proses bisnis. Jika terdapat peningkatan fasilitas produksi berakibat pada proses produksi, maka hal tersebut dapat diprediksi dengan simulasi pertumbuhan proses bisnis. Akan tetapi, pertumbuhan proses bisnis harus disesuaikan kembali dengan kebutuhan *enterprise*. Contoh kebutuhan *enterprise* yang menjadi pemicu tumbuhnya proses bisnis pada studi kasus Pondok Pesantren yaitu pertambahan jumlah santri dan kurikulum. Seiring dengan bertambahnya jumlah santri dan kurikulum, maka proses bisnis akan menjadi semakin bertambah besar. Oleh karena itu, untuk memenuhi kebutuhan tambahan *enterprise* tersebut, dapat menggunakan simulasi pertumbuhan proses bisnis. Menurut KBBI, simulasi adalah metode pelatihan yang meragakan sesuatu dalam bentuk tiruan (Kam).

Simulasi pertumbuhan proses bisnis dapat dilakukan dengan syarat proses bisnis harus *scalable*. Salah satu parameter yang digunakan untuk simulasi pertumbuhan adalah *scalability*. *Scalability* / skalabilitas yaitu potensi pertumbuhan dari sebuah proses bisnis, dapat pula disebut kemampuan suatu proses bisnis untuk tumbuh. Nilai *scalability* dapat diukur dengan perhitungan kemiripan *workflow* dan skala model proses bisnis. Nilai skala didapatkan dari hasil perhitungan beberapa parameter yaitu jumlah elemen model proses bisnis dan *Control Flow Complexity* (CFC) (Ainul, et al., 2016).

Beberapa *enterprise* mendefinisikan proses bisnis menggunakan standar BPMN (*Business Process Modelling Notation*). BPMN adalah notasi standar pemodelan diperkenalkan oleh *Open Management Group* (OMG) yang dibekali fitur dan notasi yang dapat menggambarkan proses bisnis seperti kondisi *real* (Harmon, et al., 2014). Namun, proses bisnis yang dimodelkan sebagai *workflow* menggunakan BPMN belum dapat diketahui apakah proses bisnis tersebut sudah optimal dan terbebas dari kesalahan.

Pemodelan yang umum digunakan untuk menganalisis *workflow* adalah *Petri net*. Simulasi pertumbuhan yang dilakukan pada model proses bisnis dimodelkan dengan *Petri net*, karena pemodelan menggunakan *Petri net* memudahkan pengguna dalam menganalisis *workflow*. Sehingga hasil dari simulasi pertumbuhan yaitu didapatkan variasi model proses bisnis dengan berbagai kemungkinan.

Pada penelitian ini, proses bisnis disimulasikan pertumbuhannya dengan syarat model proses bisnisnya harus *scalable*, artinya mendapatkan nilai *scalability* dari proses bisnis tersebut. Selanjutnya, model proses bisnis yang *scalable* akan ditumbuhkan menggunakan *Production Rule Cellular Automata*. Tujuan dari simulasi ini adalah untuk menumbuhkan model proses bisnis yang nilai skala proses bisnis lebih rendah dari proses bisnis yang lain sehingga didapatkan variasi proses bisnis dari berbagai kemungkinan.

Berdasarkan dari latar belakang yang di paparkan di atas, maka perlu dibuat sebuah aplikasi untuk simulasi model pertumbuhan proses yang *scalable* dan ditumbuhkan dengan *production rule* dari *Cellular Automata*.

1.2 Rumusan Masalah

1. Bagaimana membuat pola pertumbuhan *scalable business process model* pada ERP Pondok Pesantren?
2. Bagaimana mensimulasikan pertumbuhan *scalable business process model* pada ERP Pondok Pesantren berbasis *Production Rule Cellular Automata*?
3. Bagaimana mengukur ketepatan pola pertumbuhan *scalable business process model* pada ERP Pondok Pesantren?

1.3 Tujuan Penelitian

1. Membuat pola pertumbuhan *scalable business process model* pada ERP Pondok Pesantren.
2. Mensimulasikan pertumbuhan *scalable business process model* pada ERP Pondok Pesantren berbasis *Production Rule Cellular Automata.*
3. Mengukur ketepatan pola pertumbuhan *scalable business process model* pada ERP Pondok Pesantren.

1.4 Batasan Masalah

Untuk menghindari permasalahan yang ada, maka batasan masalah pada penelitian ini sebagai berikut:

1. Data uji menggunakan proses bisnis pada Pondok Pesantren yang dimodelkan dalam bentuk *Petri Net Markup Language* (PNML).
2. Proses bisnis yang akan diolah dari empat tipe Pondok Pesantren yaitu: *salaf*, *modern*, *hafidz* dan mahasiswa. Lima bagian di dalam Pondok Pesantren tersebut antara lain: Akademik, Penerimaan Santri Baru, Kesantrian, Sarpras dan Kepegawaian.

3. *Tool* yang digunakan untuk membuat model proses bisnis dan menguji *soundness* adalah WOPED.
4. Logika OR-*split* tidak digunakan dalam perhitungan *complexity* model proses bisnis ERP Pondok Pesantren.
5. Bahasa pemrograman yang digunakan untuk membangun aplikasi simulasi pertumbuhan adalah JAVA.
6. Pengujian dilakukan dengan pembobotan percabangan dan *sequence*.
7. Pengujian pada pembobotan percabangan tidak dilakukan pada logika XOR.

1.5 Manfaat Penelitian

1. Dapat mensimulasikan pertumbuhan proses bisnis berdasarkan nilai skala yang didapatkan dari tiap model proses.
2. Mengurangi waktu dalam pembuatan model proses bisnis.
3. Dapat memetakan model proses bisnis berdasarkan hasil sebelum dan sesudah di simulasikan pertumbuhannya

BAB 2

KAJIAN PUSTAKA

2.1 Enterprise Resource Planning Pondok Pesantren

Menurut Wijaya dan Darudito, *Enterprise Resource Planning* (ERP) merupakan singkatan dari tiga elemen kata yaitu terdiri dari *Enterprise* (Perusahaan), *Resource* (Sumber Daya) dan *Planning* (Perencanaan). Ketiga kata tersebut merepresentasikan sebuah konsep yang berujung pada kata terakhir yaitu *Planning* (Wijaya, et al., 2009). Dengan demikian, ERP menekankan pada aspek perencanaan. Integrasi di dalam konsep sistem ERP berhubungan yang dengan interpretasi adalah sebagai berikut:

1. Menghubungkan antara berbagai aliran proses bisnis.
2. Metode dan teknik berkomunikasi.
3. Keselarasan dan sinkronisasi operasi bisnis.
4. Koordinasi operasi bisnis.

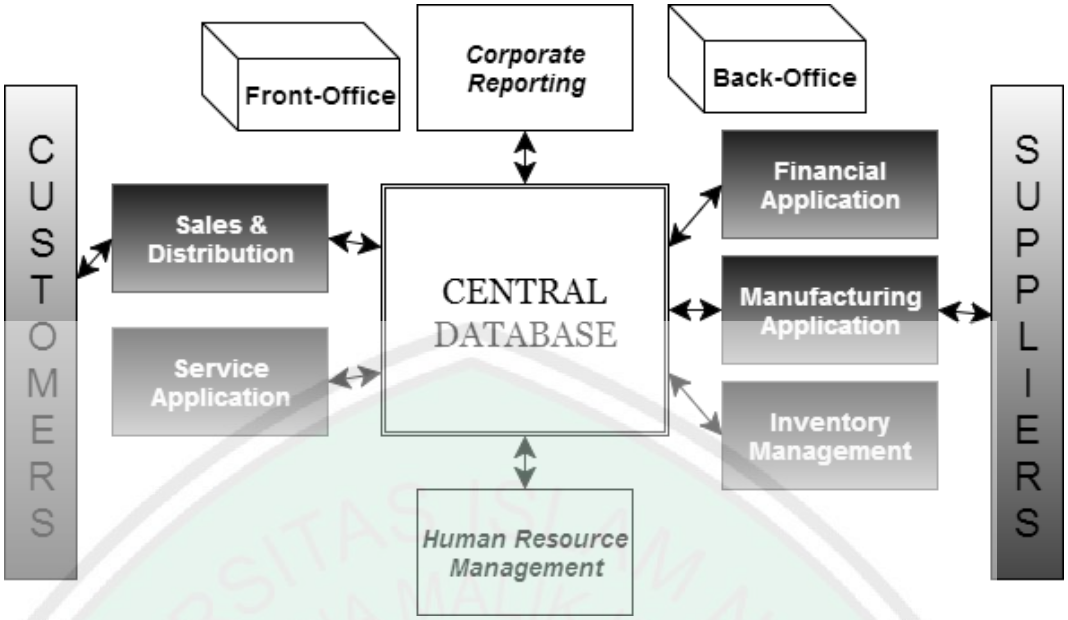
Enterprise Resource Planning (ERP) merupakan konsep untuk merencanakan dan mengelola sumber daya perusahaan, yaitu berupa paket aplikasi program terintegrasi dan multifungsi yang dirancang untuk melayani dan mendukung berbagai fungsi di dalam perusahaan, sehingga pekerjaan menjadi lebih efisien dan dapat memberikan pelayanan lebih baik bagi konsumen. Keuntungan lain dari penggunaan ERP yaitu dapat menghasilkan nilai tambah dan memberikan keuntungan maksimal bagi semua pihak yang berkepentingan (*stakeholder*).

Lembaga pendidikan Islam yang merupakan subkultural masyarakat Indonesia adalah Pondok Pesantren. Pondok Pesantren yaitu salah satu institusi unik dengan ciri khas yang kuat. Pondok Pesantren dapat juga dianalogikan sebagai *enterprise*,

oleh karena itu, proses bisnis yang terdapat pada Pondok Pesantren membutuhkan adanya perencanaan yang baik dan tepat sehingga untuk mengelolanya diperlukan ERP. ERP Pondok Pesantren membutuhkan pemodelan yang berguna untuk mendeskripsikan bagaimana alur sistem pada pondok tersebut (Fajarivan, 2017). Salah satu contoh proses bisnis yang terdapat pada pondok yaitu proses pendaftaran santri baru atau PSB. Proses PSB yang mana tidak hanya berkaitan dengan santri saja, namun berkaitan dengan keuangan, kepegawaian, kesantrian dan lain-lain. Menurut Dewanto dan Falahah konsep-konsep dasar ERP (Dewanto, et al.) yaitu:

1. ERP terdiri atas paket *software* komersial yang merupakan integrasi dari aliran informasi pada perusahaan meliputi keuangan, akuntansi, sumber daya manusia, rantai pasok dan informasi konsumen.
2. Sistem ERP adalah paket sistem informasi yang dapat dikonfigurasi dengan mengintegrasikan informasi dan proses lintas area fungsional yang terdapat dalam sebuah organisasi.
3. "Satu basis data, satu aplikasi, dan satu kesatuan antarmuka di seluruh *enterprise*".

Konsep-konsep utama ERP dapat digambarkan dalam satu diagram, oleh Devenport (Dewanto, et al.), seperti pada **Gambar 2.1** yang ditunjukkan di bawah ini :



Gambar 2.1 Konsep Dasar ERP

ERP tidak selalu mengacu pada sebuah perangkat lunak. ERP terdiri dari terminologi yang mencakup perencanaan sumber daya sebagai tuntutan bisnis, yang nantinya akan diimplementasikan dalam perangkat lunak pemrosesan yang sering disebut aplikasi ERP.

2.2 Pemodelan Proses Bisnis

Business Process Modelling (BPM) atau Pemodelan Proses Bisnis merupakan diagram umum yang mewakili urutan kegiatan. Menurut Puspa Dewi, pemodelan proses bisnis merupakan cara untuk mendesain, menganalisa dan memahami proses bisnis dengan manfaat membantu *enterprise* memahami proses bisnis, mengidentifikasi masalah yang mungkin terjadi, mendokumentasikan serta mengkomunikasikan kepada semua *stakeholder* (Puspa Dewi, et al., 2010). Kompleksitas proses bisnis membuat beberapa *enterprise* mencari cara untuk menggambarkan proses bisnis. Alternatif yang digunakan adalah dengan pemodelan proses bisnis untuk mengevaluasi, analisa dan melakukan perbaikan

proses bisnis untuk di masa mendatang. Analisa proses bisnis melibatkan proses dan sub proses di dalam aktifitas. Analisa dilakukan melalui pemodelan proses bisnis dengan menggambarkan pihak-pihak yang saling berinteraksi di dalam sistem dan dijelaskan dengan standar tertentu. Pemodelan proses bisnis dapat divisualisasikan menggunakan *flowchart* urutan kegiatan dengan dasar aturan yang relevan pada data dalam proses. Pada penelitian ini, pemodelan proses bisnis yang digunakan adalah *petri net markup language* (PNML).

2.2.1 Petri Net Markup Language (PNML)

Petri net merupakan salah satu *tool* yang menggunakan bahasa grafis dan matematis untuk pemodelan dan menganalisis proses bisnis dimana variabel-variabel hanya ada pada dua keadaan seperti aktif atau tidak, sehingga informasi dan struktur di dalamnya dapat dimodelkan (Peterson, 1981). *Petri net* dapat menggambarkan distribusi dan redistribusi yang terjadi di dalam sistem. Menurut Anggrainingsih, *Petri net* dibuat oleh ilmuan bernama Carl Adam Petri tahun 1962 sebagai alat pemodelan dari suatu proses (Rini, 2014). *Petri net* mempunyai tiga elemen di dalam pemodelan yaitu: *place*, *transition* dan *arc*. Berikut elemen yang terdapat di dalam *Petri net* sebagai berikut:

1. *Place*, digambarkan dengan bentuk bulat yang memuat informasi mengenai *event*, merepesentasikan kondisi. *Place* dapat berfungsi sebagai *input* atau *output* suatu transisi. *Place* sebagai *input* menyatakan keadaan harus terpenuhi sehingga transisi dapat terjadi.
2. *Transition*, digambarkan dengan kotak persegi panjang, merepresentasikan *event*.

3. *Arc*, digunakan untuk menghubungkan *place* ke transisi atau sebaliknya. *Arc* tidak mungkin berada diantara *place* dan *place* atau diantara *transition* dan *transition*.

Place yang dihubungkan oleh *arc* mengarah ke *transition* disebut *input* transisi. Sedangkan, *transition* yang dihubungkan oleh *arc* mengarah ke *place* disebut *output* transisi. Di dalam *place* terkadang berisi *token*, *token* didalam *place* disebut juga *marking*. *Place*, *transition*, *arc* dan *token* dalam *Petri net* dapat dinotasikan seperti **Gambar 2.2** berikut ini:

Gambar 2.2 Notasi *Petri net*

2.3 Skalabilitas Proses Bisnis

Skalabilitas proses bisnis atau *scalability* adalah efisiensi menangani pertumbuhan proses bisnis yang semakin komplek dengan pertimbangan dapat meminimalisir biaya yang akan dikeluarkan serta meningkatkan kinerja dari hasil analisis proses bisnis. *Scalability* dapat diukur dengan melibatkan beberapa parameter yaitu: *structural similarity*, *behavioral similarity*, skala model, kompleksitas dan jumlah elemen dalam proses bisnis (Ainul, et al., 2016). Skala model didapatkan dengan perhitungan pada rumus persamaan 2.1 di bawah ini :

$$Skala(A) = E(A) * CFC(A) \quad \text{.....(2.1)}$$

Keterangan:
Skala (A) = Nilai skala model A
E (A) = Jumlah elemen dalam model proses bisnis A
CFC (A) = Nilai *Control Flow Complexity* model proses bisnis A

Perhitungan tersebut juga berlaku kepada semua proses bisnis yang nantinya dihitung nilai *scalability*-nya. Berikut perhitungan CFC ditunjukkan pada rumus persamaan 2.2 di bawah ini:

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

(2.2)

Dimana diturunkan dari persamaan di bawah ini:

$$CFC_{XOR-split}(A) = fanout(A) \text{..... (2.3)}$$

$$CFC_{OR-split}(A) = 2^{fanout(A)} - 1 \text{..... (2.4)}$$

$$CFC_{OR-split}(A) = 2^{fanout(A)} - 1 \text{ ..... (2.5)}$$

Jika nilai *Control Flow Complexity* (CFC) semakin besar maka tingkat kompleksitas model proses bisnis semakin tinggi (Rolón, et al., 2009). Selanjutnya, dilakukan perhitungan perbandingan skala antar proses bisnis yang ditunjukkan pada rumus persamaan 2.6 sebagai berikut :

$$skala(A,B) = \frac{skala(A)}{skala(B)} \text{..... (2.6)}$$

Keterangan :
Skala (A,B) = Nilai skala perbandingan antar model proses bisnis A dan B

Kemudian untuk mendapatkan nilai *scalability*, kalikan nilai *similarity worklfow* antara model proses bisnis A dengan B. Sehingga didapatkan formula seperti pada persamaan 2.7 di bawah ini :

$$scalability(A,B) = 1 - \big(skala(A,B) * sim(A,B)\big) \text{.....(2.7)}$$

Keterangan :
Sim (A, B) = Nilai *similarity* model proses bisnis A dengan B

Berdasarkan dengan persamaan 2.7 di atas, didapatkan nilai *scalability* dari model proses bisnis.

2.3.1 *Similarity Workflow*

Berdasarkan hasil penelitian Arif Wahyu, perhitungan *similarity workflow* adalah kegiatan perhitungan dengan proses membandingkan beberapa *workflow* dengan memeriksa apakah terdapat kesamaan antar satu *workflow* dengan lainnya. Perhitungan *similarity workflow* dapat dihitung dengan dua aspek berbeda yaitu aspek *structural* dan aspek *behavioral* (Prasetya, 2017).

1. *Structural Similarity*

   Structural similarity atau similaritas struktural adalah menghitung kemiripan struktur dari dua buah *graph*. *Structural similarity* dihitung dengan metode *Jaccard coefficient* (Ainul, et al., 2016) yang digunakan untuk membandingkan interseksi dan gabungan dari dua *graph* yang dirumuskan pada persamaan 2.8 sebagai berikut:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \quad \text{..... (2.8)}$$

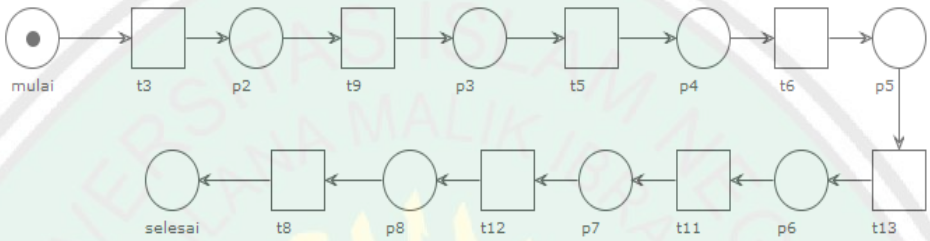
Keterangan:
J (A, B) = Nilai *Jaccard* untuk model proses bisnis A dan B
$A \cap B$ = Nilai intersek dari model proses bisnis A dan B
$A \cup B$ = Nilai gabungan dari model proses bisnis A dan B

Rumus *Jaccard coefficient* pada persamaan 2.8 di atas, diturunkan pada persamaan 2.9 seperti di bawah ini:

$$J(A,B) = \frac{E(A)}{(E(A)+E(B))-E(A)} \quad \text{..... (2.9)}$$

Keterangan:
J (A, B) = Nilai Jaccard untuk model proses bisnis A dan B
A = Model proses bisnis A
B = Model proses bisnis B
E = Jumlah elemen di dalam proses bisnis

Untuk menghitung elemen pada *structural similarity* ditunjukkan pada **Gambar 2.3** seperti di bawah ini:



Gambar 2.3 Contoh Model *Petri net*

Structural similarity, jumlah elemen adalah mulai, t3, p2, t9, p3, t5, p4, t6, p5, t13, p6, t11, p7, t12, p8, t8, selesai, mulai-t3 , t3-p2, p2-t9, t9-p3, p3-t5, t5-p4, p4-t6, t6-p5, p5-t13, t13-p6, p6-t11, t11-p7, p7-t12, t12-p8, p8-t8, t8-selesai = 33 elemen.

2. *Behavioral Similarity*

Behavioral similarity atau similaritas perilaku adalah perhitungan kesamaan perilaku dari dua *graph*, atau berdasarkan perilaku antar proses bisnis. Nilai similaritas perilaku didapatkan dengan relasi antar aktifitas pada proses bisnis. Metode yang digunakan untuk menghitung similaritas perilaku sama dengan pada similaritas struktural. Tetapi terdapat perbedaan pada menghitung jumlah elemen pada model proses bisnis yaitu berdasarkan pada **Gambar 2.3**. *Behavioral similarity*, jumlah elemen adalah t3-t9, t9-t5, t5-t6, t6-t13, t13-t11, t11-t12, t12-t8 = 7 elemen.

2.4 Simulasi Pertumbuhan Proses Bisnis

Proses bisnis yang dirancang mungkin bersifat fungsional, tetapi sebagian besar tidak optimal untuk *enterprise* sehingga dapat di atasi dengan menjalankan simulasi pertumbuhan proses bisnis. Simulasi pertumbuhan proses bisnis membantu *enterprise* dengan melakukan analisa dan pemahaman lebih lanjut mengenai proses bisnis. Tujuan dari simulasi pertumbuhan proses bisnis adalah optimasi untuk meningkatkan kinerja dan pengurangan biaya selaras dengan semakin kompleksnya proses bisnis. Strategi simulasi lainnya adalah membantu mengembangkan alur proses bisnis dengan menyediakan *platform* serta memfasilitasi tentang faktor analisis proses bisnis yang efisien. Simulasi terkait dengan *scalability* yaitu menghitung nilai skala dari proses bisnis sehingga dapat meminimalisir biaya serta meningkatkan hasil yang di inginkan berdasarkan proses bisnis yang disimulasi. Perhitungan *scalability* dan *similarity workflow* diperlukan untuk melihat simulasi pertumbuhan proses bisnis apakah berjalan dengan baik atau tidak.

2.4.1 Pertumbuhan Proses Bisnis

Pertumbuhan proses bisnis dapat digunakan untuk menumbuhkan proses bisnis agar menghasilkan variasi proses bisnis dengan berbagai kemungkinan. Simulasi pertumbuhan membutuhkan beberapa aturan untuk menumbuhkan proses bisnis itu. Ada teori *L-System* yang digunakan oleh beberapa peneliti untuk simulasi pertumbuhan tanaman. *L-System* sendiri merupakan *grammar* atau pola aturan pada tanaman yang memiliki beberapa pola kesamaan dan bersifat berulang (Suhartono, 2012). Simulasi pertumbuhan tanaman direpresentasikan ke dalam simulasi pertumbuhan suatu graph. Oleh karena itu, teori *L-System* mungkin akan tepat jika

digunakan untuk simulasi pertumbuhan proses bisnis. Namun, sebelum adanya teori *L-System* terdapat Teori *Production Rule Cellular Automata*, yang nantinya akan digunakan dalam penelitian ini.

1. *Production Rule Cellular Automata*

Production Rule Cellular Automata merupakan bagian di dalam *Cellular Automata*. *Cellular Automata* adalah model diskrit pada ilmu komputer, matematika, fisika, ilmu kompleksitas, biologi teoritis dan pemodelan *Cellular Automata* juga disebut runag *cellular*, *tessellation automata*, struktur homogen, struktur *cellular*, struktur tesselation dan *array* yang berulang-ulang (Wolfram, 1983). *Cellular Automata* terdiri dari beberapa *grid* sel, masing-masing pada *finite state grid* sel dibatasi (seperti pada map). *Grid* sel dapat dibatasi dengan dimensi yang terbatas. Untuk setiap sel, kumpulan sel yang ada disekitarnya disebut tetangga atau sel relatif yang didefinisikan. Inisial state (waktu t = 0) dipilih sebagai keadaan awal setiap sel. Generasi dibuat (t menjadi = 1), yang disebut sebagai aturan produksi dari *Cellular Automata* (H Margoulus, et al., 1987). Sedangkan perbedaan dengan teori *automata* adalah studi mengenai mesin abstrak (lebih tepatnya mesin 'matematika' abstrak atau sistem) dan dalam komputasi dapat diselesaikan dengan mesin ini. Mesin abstrak ini disebut *automata*. *Automata* berasal dari bahasa Yunani (Αυτόματα) yang berarti sesuatu melakukan sesuatu itu sendiri. Teori *automata* juga terkait dengan bahasa formal (Hopcroft, et al., 2006), dan sebagai *automata* sering diklasifikasikan oleh kelas bahasa formal yang dikenali. *Automata* berkaitan erat dengan teori bahasa formal, dicontohkan dengan suatu kalimat dengan menerapkan serangkaian aturan produksi pada sebuah simbol "akar". Proses penerapan aturan produksi

dapat digambarakan sebagai suatu diagram pohon. Dan juga terdapat beberapa aturan produksi *Cellular Automata* atau dapat disebut *Production Rule Cellular Automata* antara lain:

a. Aturan produksi $\alpha \rightarrow \beta$ yang diterapkan pada suatu *string* $w = a\alpha c$ mengganti kemunculan α menjadi β sehingga string tersebut menjadi $w = a\beta c$ sehingga daat ditulis $a\alpha c \Rightarrow a\beta c$ $(a\alpha c\ memproduksi\ a\beta c)$.

b. Produksi tersebut dapat diterapkan berkali-kali

$$w_1 \Rightarrow w_2 \Rightarrow w_3 \Rightarrow w_n \ldots\ldots\ldots (2.10)$$

c. Atau dapat ditulis $w_1 \Rightarrow^* w_n$, jika minimal harus ada 1 aturan produksi yang ditetapkan : $w_1 \Rightarrow^+ w_n$

2.5 PenelitianTerkait

Berikut daftar beberapa penelitian terkait dengan penelitian yang akan dilakukan seperti pada **Tabel 2.1** di bawah ini:

Tabel 2.1 *Related Work*

Identitas Penelitian	Masalah	Metode Penyelesaian	Hasil Penelitian
Belgacem Ben Youssef, gang Cheng, Kyriacos Zygourakis and Pauline Markenscoff.”Parallel Implementation of A Cellular *Automata* Modeling The Growth Of Three-Dimensional Tissues”. *The International Journal of High Performance Computing Applications Volume 21, No2, pp. 196-209*.Summer.2007.	1)Pendekatan untuk mengobati jaringan atau organ yang melibatkan pada jaringan *bioartifial*. 2)Rekayasa pengganti jaringan yang relatif mahal dengan cara manual.	Model komputasi menggunakan metode *cellular automata* untuk menggambarkan perilaku dinamis dari populasi sel yang bermigrasi. Serta tiga algoritma paralel dikembangkan untuk mendekati algoritma *sequntial* yang menggambarkan pertumbuhan jaringan.	Implemetasi paralel model baru yang secara akurat menggambarkan dinamika rumit populasi besar sel yang bermigrasi. Simulasi dilakukan dengan three-dimesional Cellular *Automata*.
Wael Sellami, Hatem Hadj Kecem, and Ahmed Hadj Kacem.”Controlling Elasticity Dependencies	Elastisitas proses bisnis dan Saas pada tingkat independen mengakibatkan pemborosan sumber	1) Melakukan pendekatan holistik untuk secara dinamis mengontrol	Menghasilkan algoritma untuk mengontrol dan menganalisa elastisitas yang

Identitas Penelitian	Masalah	Metode Penyelesaian	Hasil Penelitian
For Multi-tenant Business Process". *2015 IEEE 12th International Conference on e-Business Engineering*. *IEEE*, 2015	daya yang siginifikan berkaitan dengan skalabilitas.	mekanismne elastisitas dependensi fungsional antar proses *multi-tenant.* 2) Menerapkan algoritma *auto-scaling* dan secara dinamik dapat mengatur mekanisme dependensi kontrol elatisitas.	berdasar pada pola pelayanan untuk manajemen elastisitas.
Sergio Hernandez, Joaquin Ezpeleta."Assesing Process Discovery Scalability in Data Intensive Environments". *2015 IEEE/ACM And International Symposium on Big Data Computing*. *IEEE*, 2015.	Fenomena ledakan data yang berdampak pada proses bisnis. Pada proses *discovery*, mengatur secara otomatis model proses dari *event data* berkaitan dengan eksekusi proses bisnis.	1) Menilai pengaplikasian *scalability* pada teknik proses *discovery* pada data *intensive environment.* 2) Mengkomputasi abstraksi internal data dengan teknik *discovery* menggunakan *framework MapReduce.*	Mengevaluasi skalabilitas dan validitas menggunakan pendekatan berbasis *MapReduce*. Pendekatan didasarkan pada komputasi abstraksi yang mampu menangani *event log* yang besar secara efisien dan terukur.
Muhammad Ainul Yaqin, Riyanarto Sarno and Abd. Charis Fauzan."Scalability Measurement of Business Process Model using Business Processes Similarity and Complexity".*Informatic Department, Institut Teknologi Sepuluh Nopember*. 2016	Untuk mendapatkan nilai skalalabititas proses bisnis yang nantinya mampu menangani pertumbuhan jumlah proses atau potensi proses bisnis jika diperbesar.	Mengukur skalabilitas dengan cara membandingkan struktur dan perilaku proses bisnis serta membandingkan kompleksitasnya.	Didapatkan model proses bisnis dengan indeks kemiripan yang besar akan menghasilkan nilai skalabilitas yang besar juga.

Penelitian yang dilakukan penulis yaitu simulasi pertumbuhan proses bisnis pada ERP Pondok Pesantren menggunakan teori *Production Rule Cellular Automata*. Aturan produksi tersebut diterapkan pada *proposed algorithm* yang berupa *flowchart* algoritma dari penelitian ini. Perbedaan penelitian yang akan dilakukan dengan penelitian terkait yang telah dijelaskan pada **Tabel 2.1** diatas,

terletak pada proses menumbuhkan proses bisnis. Menumbuhkan proses bisnis maksudnya adalah menumbuhkan elemen baru pada proses bisnis yang akan ditumbuhkan, sehingga dapat diketahui variasi proses bisnis dari berbagai kemungkinan.

Pertumbuhan proses bisnis tanpa menggunakan metode akan mengalami kesulitan, sehingga digunakan teori *Production Rule Cellular Automata* yang mempermudah proses pertumbuhan proses bisnis. Allah SWT telah berfirman dalam QS. Al-Insyirah ayat 5-8 bahwa segala kesukaran atau kesulitan suatu urusan, pasti ada kelapangan yakni kemudahan. Karena Allah menyertakan suatu kesulitan pastilah juga sudah menyiapkan solusi untuk menyelesaikannya.

فَإِنَّ مَعَ الْعُسْرِ يُسْرًا

Karena sesungguhnya sesudah kesulitan itu ada kemudahan (QS. Al-Insyirah : 5)

إِنَّ مَعَ الْعُسْرِ يُسْرًا

Sesungguhnya sesudah kesulitan itu ada kemudahan. (QS. Al-Insyirah : 6)

فَإِذَا فَرَغْتَ فَانْصَبْ

Maka apabila kamu telah selesai (dari sesuatu urusan), kerjakanlah dengan sungguh-sungguh (urusan) yang lain. (QS. Al-Insyirah : 7)

وَإِلَىٰ رَبِّكَ فَارْغَبْ

dan hanya kepada Tuhanmulah hendaknya kamu berharap. (QS. Al-Insyirah : 8)

Menurut tafsir *Jalalayn* dari surat Al-Insyirah ayat 5-8: Karena sesungguhnya sesudah kesulitan itu atau kesukaran itu (ada kelapangan) yakni kemudahan. (Sesungguhnya sesudah kesulitan itu ada kelapangan), Nabi SAW banyak sekali mengalami kesulitan dan hambatan dari orang-orang kafir, kemudian beliau

mendapatkan kelapangan dan kemudahan, yaitu setelah beliau mengalami kemenangan atas mereka. (Maka apabila kamu telah selesai) dari salat (bersungguh-sungguhlah kamu) di dalam berdoa. (Dan hanya kepada Rabbmulah hendaknya kamu berharap) atau meminta dengan merendahkan diri.

BAB 3

METODE PENELITIAN

3.1 Desain Penelitian

3.1.1 Gambaran Umum

Aplikasi yang akan dibangun adalah aplikasi untuk mensimulasikan pertumbuhan *scalable business process model* artinya proses bisnis yang akan ditumbuhkan harus *scalable*. Selanjutnya proses bisnis yang *scalable* disimulasikan pertumbuhannya sehingga didapatkan beberapa variasi proses bisnis dari berbagai kemungkinan. Pertumbuhan proses bisnis ditandai dengan penurunan nilai *scalability* dari model proses bisnis. Teori yang digunakan untuk mensimulasikan pertumbuhan proses bisnis yaitu *Production Rule Cellular Automata*. Model proses bisnis yang digunakan diperoleh dari empat tipe Pondok Pesantren dan dimodelkan dengan *Petri net Markup Language* (PNML).

3.1.2 Sumber Data

Terdapat dua tipe sumber data yang digunakan dalam penelitian ini sebagai berikut:

1. Data Primer

   Data primer diperoleh dengan melakukan pengumpulan data ke lokasi penelitian. Data yang digunakan dalam penelitian ini merupakan proses bisnis dari empat tipe pondok berbeda dengan lima bagian proses yang ada dalam ERP Pondok Pesantren. Data berupa proses bisnis yang nanti akan dijadikan sebagai objek penelitian. Adapun empat tipe pondok serta lokasi penelitian yang digunakan dalam penelitian ini yaitu:

a. Tipe pondok *Salaf* (Lokasi penelitian: Pondok Salaf Anwarul Huda Malang)
b. Tipe pondok *Modern* (Lokasi penelitian: Pondok Modern Al-Rifaie 1 Gondanglegi)
c. Tipe pondok Mahasiswa (Lokasi penelitian: Pondok Pesantren Mahasiswa Syai Urrifa' Malang)
d. Tipe pondok *Tahfidz* (Lokasi penelitian: Pondok Pesantren Ar-Rohmah)

2. Data Sekunder

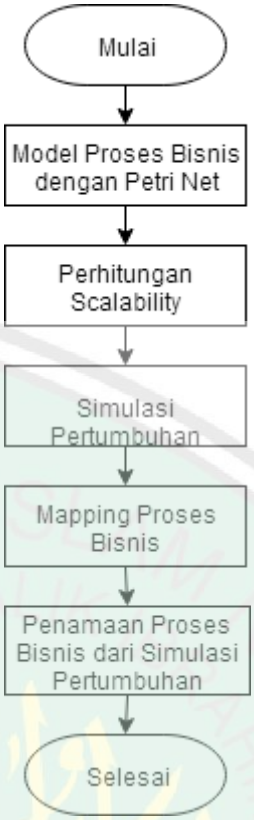
Data sekunder adalah data yang diperoleh dari penelitian terkait sebelumnya berupa ERP Pondok Pesantren (Prasetya, 2017).

3.1.3 Tipe Penelitian

Tipe penelitian yang dilakukan adalah pendekatan kuantitatif. Dikatakan kuantitatif karena berdasar pada angka perhitungan kemiripan *workflow*, *scalability* dan *Production Rule Cellular Automata*.

3.1.4 Prosedur Penelitian

Prosedur dalam penelitian ini menjelaskan bagaimana pelaksanaan penelitian. Prosedur penelitian di awali dengan tahapan pemodelan proses bisnis menggunakan *Petri net*, kemudian mensimulasikan pertumbuhan model proses bisnis, sehingga dapat dipetakan proses bisnis sebelum dan sesudah disimulasi pertumbuhannya. Prosedur penelitian digambarkan secara umum seperti pada **Gambar 3.1** di bawah ini:



Gambar 3.1 Prosedur Penelitian

3.2 Rancangan Percobaan

Pada rancangan percobaan dilakukan beberapa tahapan untuk pembangunan sistem. Rancangan percobaan pada penelitian yang akan dilakukan berdasarkan **Gambar 3.1** di prosedur penelitian yang alurnya sebagai berikut:

3.2.1 Model Proses Bisnis dengan *Petri net*

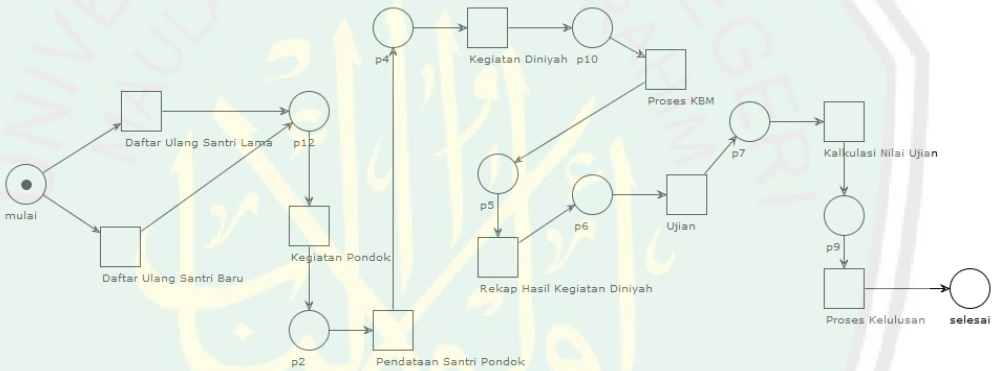
Input-an yang akan diolah dalam penelitian ini berupa *workflow*.pnml yaitu model proses bisnis yang dimodelkan menggunakan *Petri net*. Berikut proses bisnis dari empat tipe Pondok Pesantren berbeda yang mencakup lima bagian proses didalamnya. Data proses bisnis didapatkan dari observasi di empat lokasi dengan tipe pondok berbeda, yaitu Pondok Pesantren Salaf Anwarul Huda Malang (tipe pondok *salaf*), Pondok Pesantren Modern Ar-Rifaie 1 Gondanglegi (tipe

pondok *modern*), Pondok Pesantren Mahasiswa Syai Urrifa' Malang (tipe pondok mahasiswa) dan Pondok Pesantren Ar-Rohmah Malang (tipe pondok *tahfidz*) dari penelitian sebelumnya (Prasetya, 2017). Data proses bisnis yang didapatkan dari empat tipe pondok berbeda yang terdiri dari:

1. Proses Bisnis Pondok Pesantren *Salaf* Anwarul Huda Malang

   a. Bagian Akademik

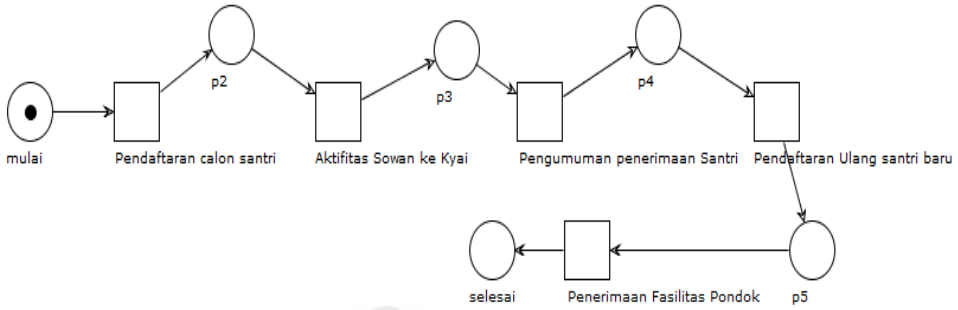
      Proses bisnis pendataan bagian akademik di Pondok Pesantren Salaf Anwarul Huda dimodelkan dengan *Petri net* ditunjukkan pada **Gambar 3.2** di bawah ini:



Gambar 3.2 Proses Bisnis Akademik *Salaf*

   b. Bagian Penerimaan Santri Baru

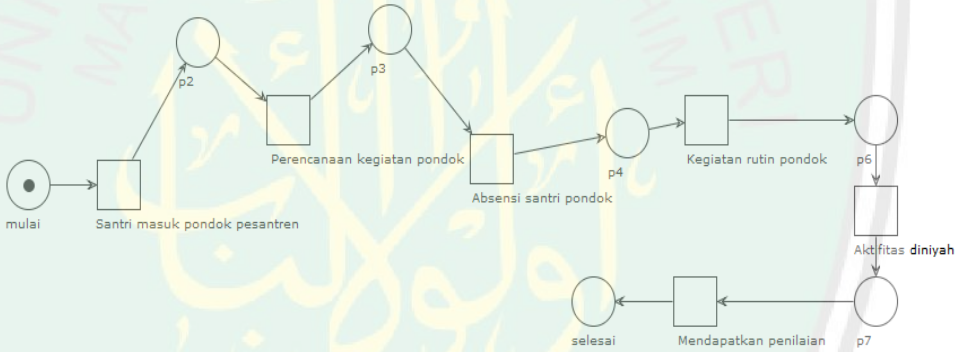
      Proses bisnis bagian Penerimaan Santri Baru (PSB) di Pondok Pesantren Salaf Anwarul Huda ditunjukkan pada **Gambar 3.3** di bawah ini:



Gambar 3.3 Proses Bisnis PSB *Salaf*

c. Bagian Kesantrian

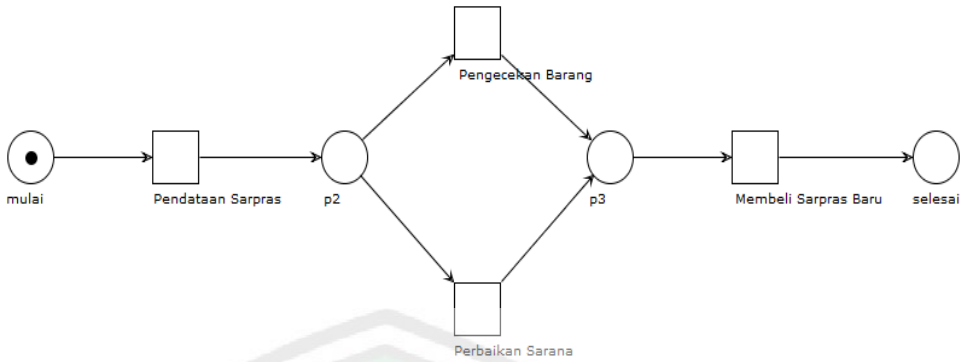
Proses bisnis bagian Kesantrian Pondok Pesantren Salaf Anwarul Huda dimodelkan dengan *Petri net* ditunjukkan pada **Gambar 3.4** seperti di bawah ini:



Gambar 3.4 Proses Bisnis Kesantrian *Salaf*

d. Bagian Sarana Prasarana

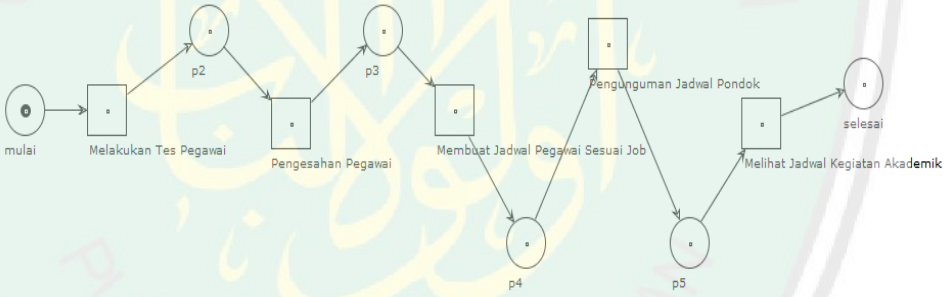
Proses bisnis bagian Sarana Prasarana Pondok Pesantren Salaf Anwarul Huda dimodelkan dengan *Petri net* ditunjukkan pada **Gambar 3.5** seperti di bawah ini:



Gambar 3.5 Proses Bisnis Sarana Prasarana *Salaf*

e. Bagian Kepegawaian

Proses bisnis bagian Kepegawaian Pondok Pesantren Salaf Anwarul Huda yang dimodelkan dengan *Petri net* ditunjukkan pada **Gambar 3.6** seperti di bawah ini:

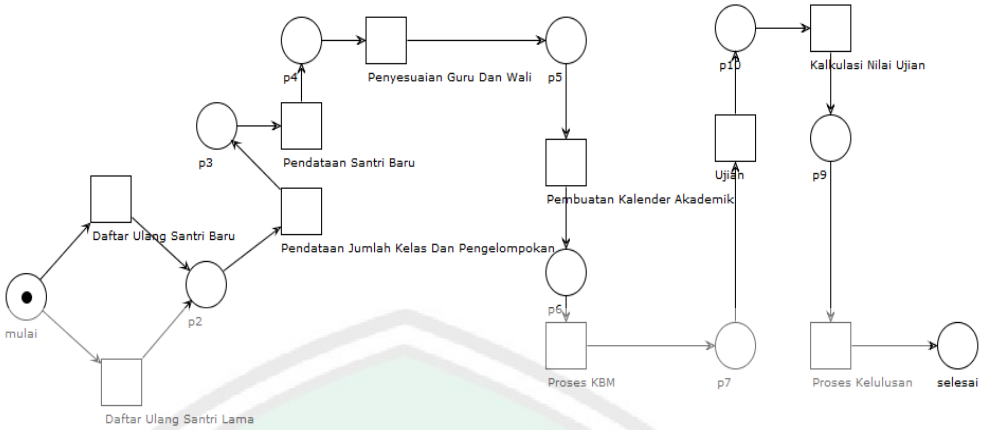


Gambar 3.6 Proses Bisnis Kepegawaian *Salaf*

2. Proses Bisnis Pondok Pesantren Modern Ar-Rifaie Gondanglegi

a. Bagian Akademik

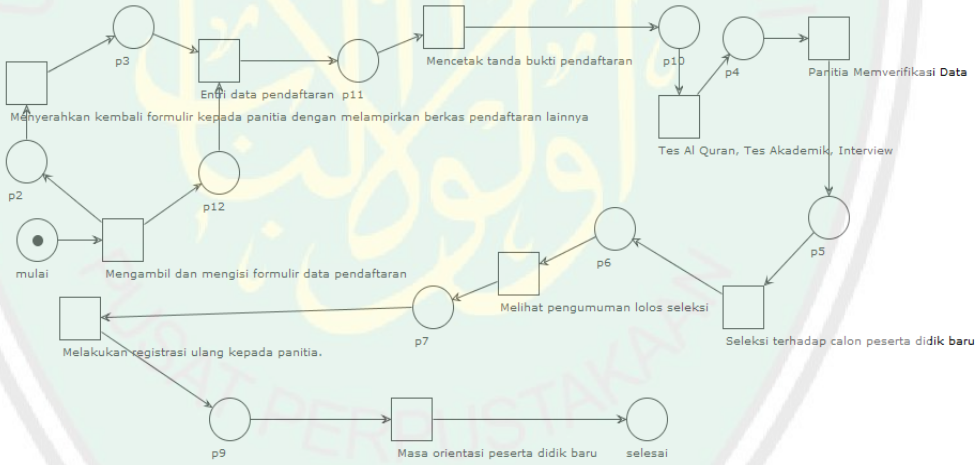
Proses bisnis bagian Akademik Pondok Pesantren Modern Ar-Rifaie Gondanglegi yang dimodelkan dengan *Petri net* ditunjukkan pada **Gambar 3.7** di bawah ini:



Gambar 3.7 Proses Bisnis Akademik *Modern*

b. Bagian Penerimaan Santri Baru

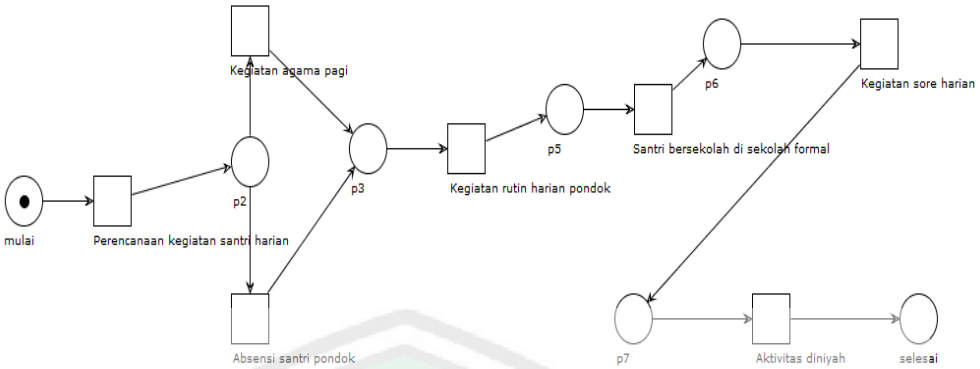
Proses bisnis bagian Penerimaan Santri Baru di Pondok Pesantren Modern Ar-Rifaie Gondanglegi yang dimodelkan dengan *Petri net* ditunjukkan pada **Gambar 3.8** seperti di bawah ini:



Gambar 3.8 Proses Bisnis PSB *Modern*

c. Bagian Kesantrian

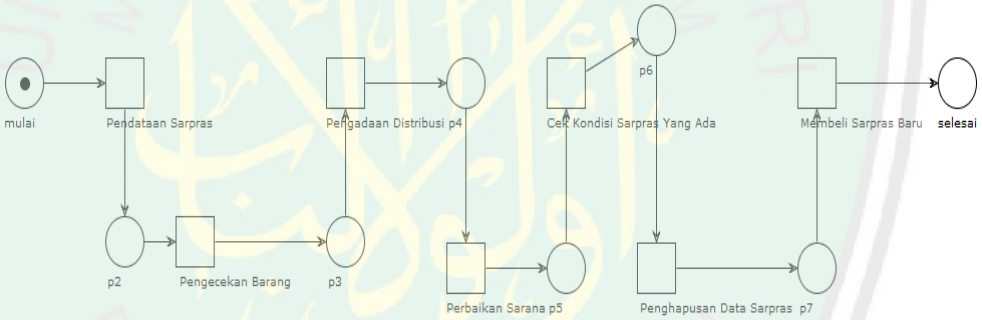
Proses bisnis bagian Kesantrian Pondok Pesantren Modern Ar-Rifaie Gondanglegi yang dimodelkan dengan *Petri net* ditunjukkan pada **Gambar 3.9** seperti di bawah ini:



Gambar 3.9 Proses Bisnis Kesantrian *Modern*

d. Bagian Sarana Prasarana

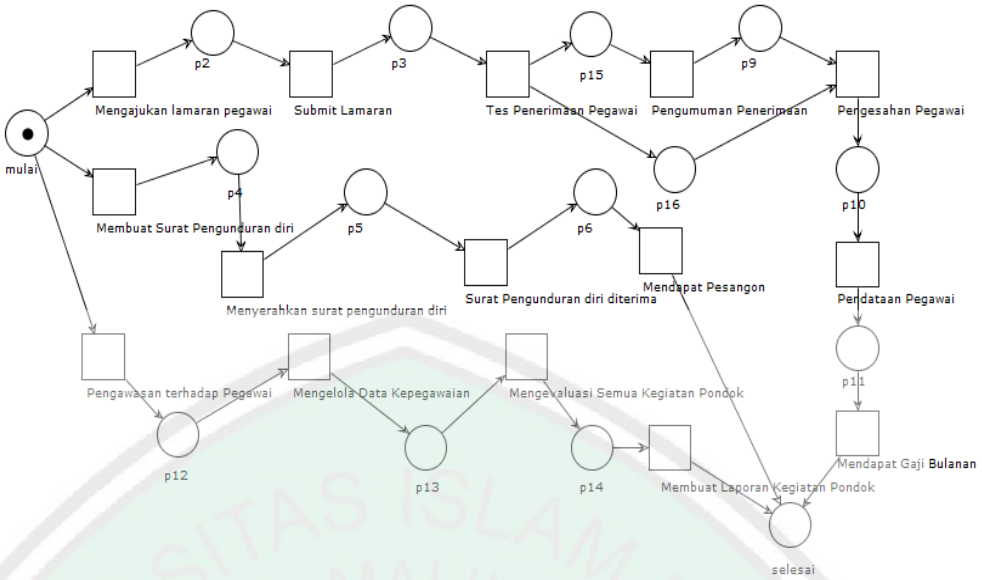
Proses bisnis bagian Sarana Prasarana di Pondok Pesantren Modern Ar-Rifaie yang dimodelkan dengan *Petri net* ditunjukkan pada **Gambar 3.10** seperti di bawah ini:



Gambar 3.10 Proses Bisnis Sarana Prasarana *Modern*

e. Bagian Kepegawaian

Proses bisnis bagian Kepegawaian di Pondok Pesantren Modern Ar-Rifaie yang dimodelkan dengan *Petri net* ditunjukkan pada **Gambar 3.11** seperti di bawah ini:

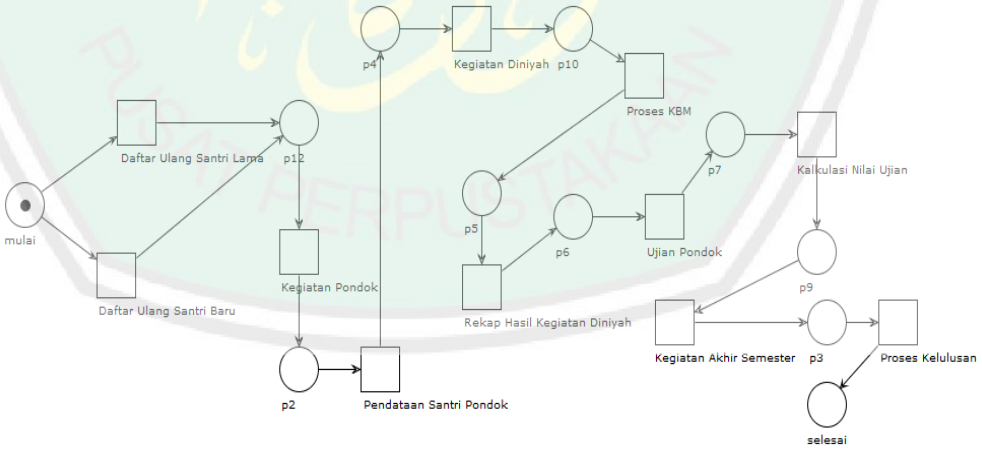


Gambar 3.11 Proses Bisnis Kepegawaian *Modern*

3. Proses Bisnis Pondok Pesantren Mahasiswa Syai Urrifa' Malang

   a. Bagian Akademik

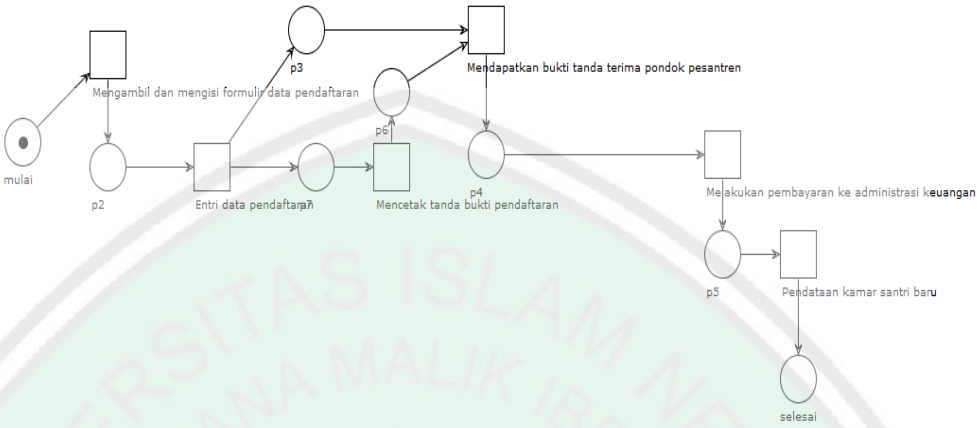
      Proses bisnis bagian Akademik di Pondok Pesantren Mahasiswa Syai Urrifa' Malang yang dimodelkan dengan *Petri net* ditunjukkan pada **Gambar 3.12** di bawah ini:



Gambar 3.12 Proses Bisnis Akademik Mahasiswa

   b. Bagian Penerimaan Santri Baru

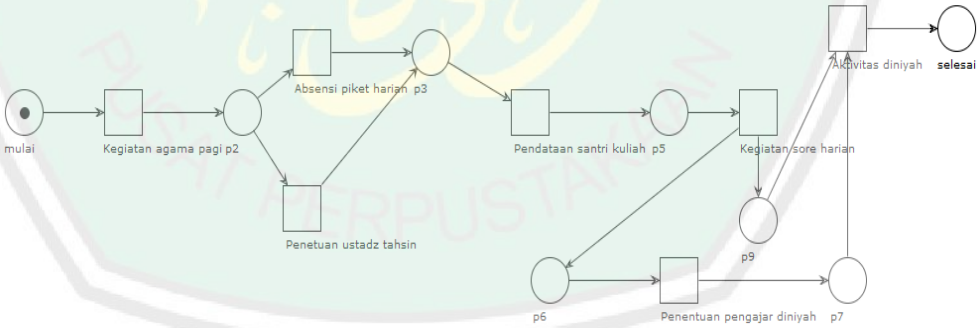
Proses bisnis bagian Penerimaan Santri Baru di Pondok Pesantren Mahasiswa Syai Urrifa' yang dimodelkan dengan *Petri net* ditunjukkan pada **Gambar 3.13** di bawah ini:



Gambar 3.13 Proses Bisnis PSB Mahasiswa

c. Bagian Kesantrian

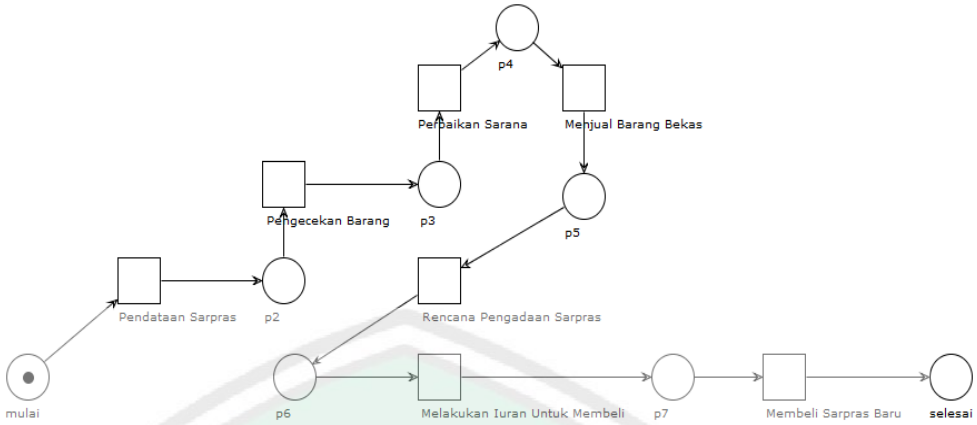
Proses bisnis bagian Kesantrian Pondok Pesantren Mahasiswa Syai Urrifa' Malang yang dimodelkan dengan *Petri net* ditunjukkan pada **Gambar 3.14** seperti di bawah ini:



Gambar 3.14 Proses Bisnis Kesantrian Mahasiswa

d. Bagian Sarana Prasarana

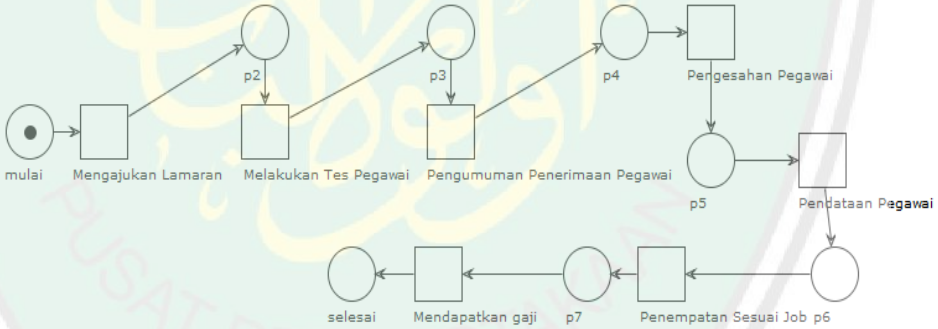
Proses bisnis pendataan Sarana dan Prasarana di Pondok Pesantren Mahasiswa Syai Urrifa' Malang yang dimodelkan dengan *Petri net* ditunjukkan pada **Gambar 3.15** seperti di bawah ini:



Gambar 3.15 Proses Bisnis Sarana Prasarana Mahasiswa

e. Bagian Kepegawaian

Proses bisnis pendataan Kepegawaian di Pondok Pesantren Mahasiswa Syai Urrifa' Malang yang dimodelkan dengan *Petri net* ditunjukkan pada **Gambar 3.16** seperti di bawah ini:

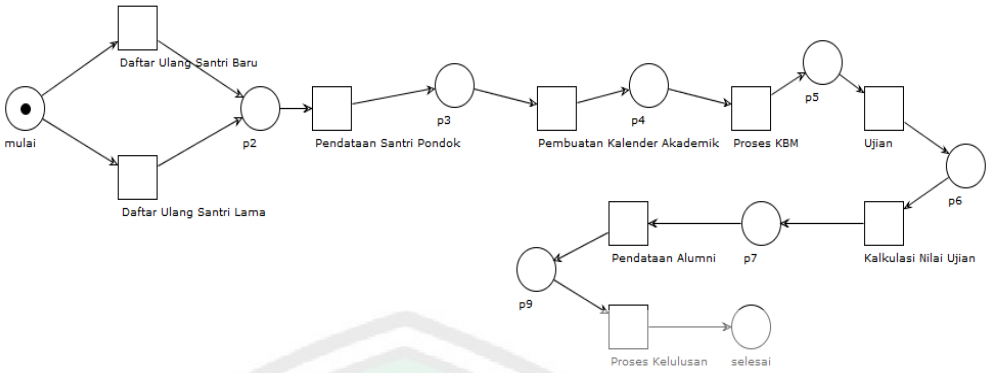


Gambar 3.16 Proses Bisnis Kepegawaian Mahasiswa

4. Proses bisnis Pondok Pesantren Ar-Rohmah

a. Bagian Akademik

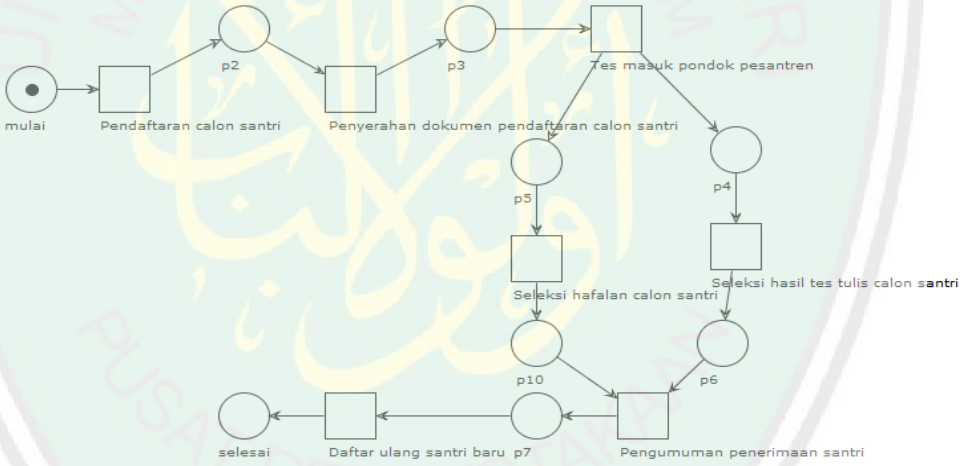
Proses bisnis pendataan Akademik Pondok Pesantren Ar-Rohmah yang dimodelkan dengan *Petri net* ditunjukkan pada **Gambar 3.17** seperti di bawah ini:



Gambar 3.17 Proses Bisnis Akademik *Tahfidz*

b. Bagian Penerimaan Santri Baru

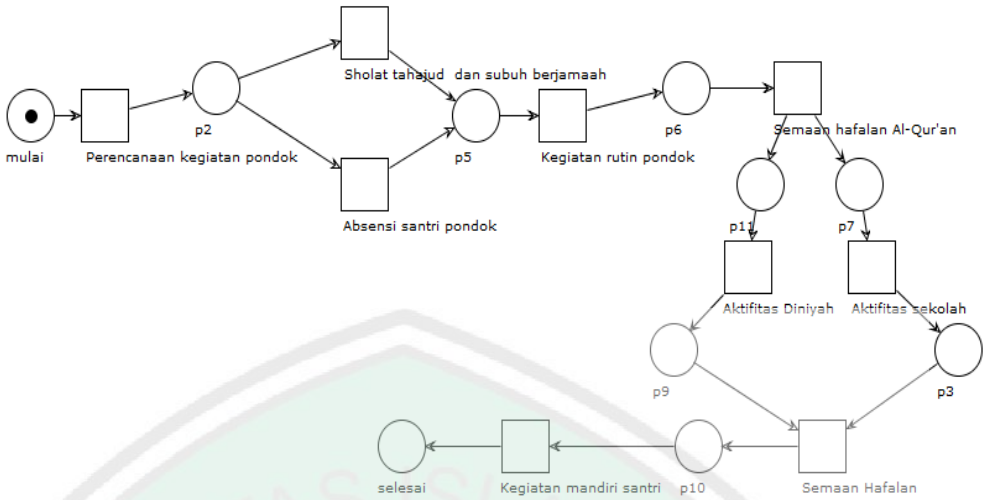
Proses bisnis bagian Penerimaan Santri Baru di Pondok Pesantren Ar-Rohmah yang dimodelkan dengan *Petri net* ditunjukkan pada **Gambar 3.18** seperti di bawah ini:



Gambar 3.18 Proses Bisnis PSB *Tahfidz*

c. Bagian Kesantrian

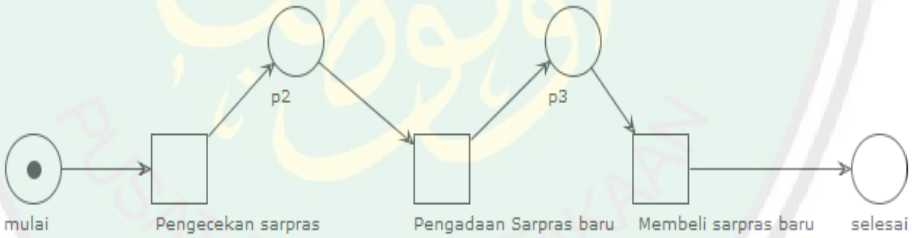
Proses bisnis bagian Kesantrian di Pondok Pesantren Ar-Rohmah yang dimodelkan dengan *Petri net* ditunjukkan pada **Gambar 3.19** seperti di bawah ini:



Gambar 3.19 Proses Bisnis Kesantrian *Tahfidz*

d. Bagian Sarana Prasarana

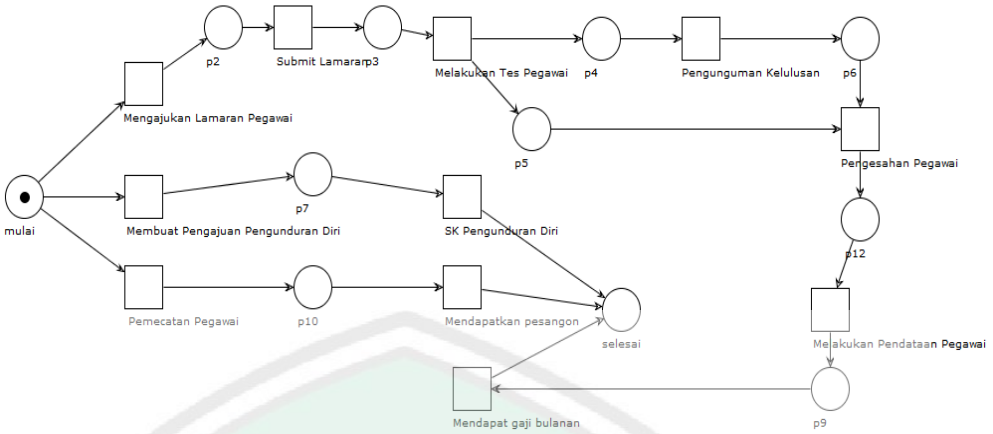
Proses bisnis bagian Sarana Prasarana Pondok Pesantren R-Rohmah yang dimodelkan dengan *Petri net* ditunjukkan pada **Gambar 3.20** seperti di bawah ini:



Gambar 3.20 Proses Bisnis Sarana Prasarana *Tahfidz*

e. Bagian Kepegawaian

Proses bisnis pendataan bagian Kepegawaian di Pondok Pesantren Ar-Rohmah yang dimodelkan dengan *Petri net* ditunjukkan pada **Gambar 3.21** seperti di bawah ini:



Gambar 3.21 Proses Bisnis Kepegawaian *Tahfidz*

3.2.2 Perhitungan *Scalability*

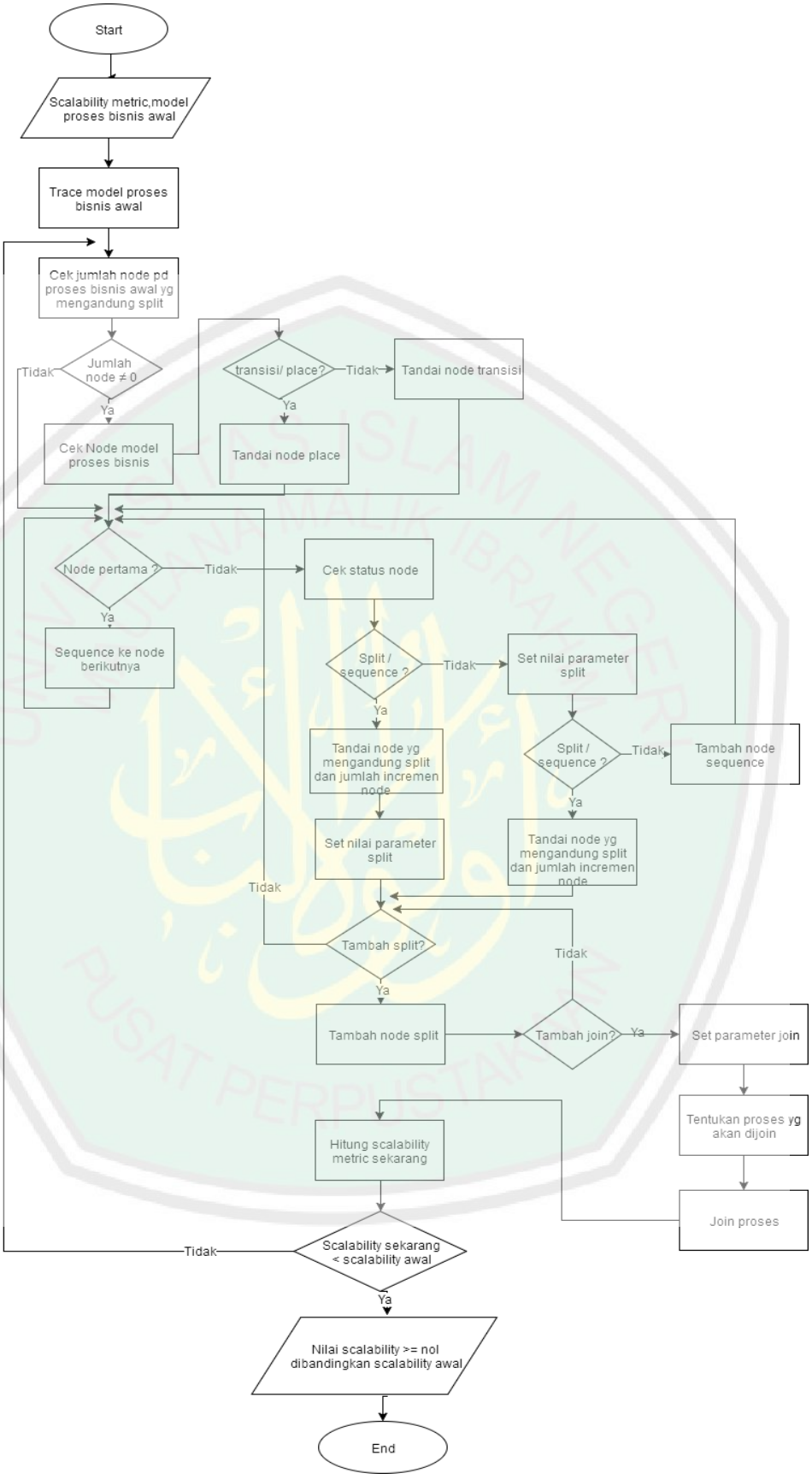
Setelah model proses bisnis di*input*kan, selajutnya dihitung *scalability*-nya. Nilai *scalability* dapat diukur dengan melibatkan beberapa parameter yaitu: *structural similarity*, *behavioral similarity*, skala model, kompleksitas dan jumlah elemen dalam model proses bisnis (Ainul, et al., 2016). Rumus s*calabil*ity di bawah ini sama seperti pada persamaan 2.7 yaitu:

$$scalability(A,B) = 1 - \big(skala(A,B) * sim(A,B)\big)$$

Keterangan:
sim (A, B) = Nilai *similarity* model proses bisnis A dengan B
skala (A, B) = Nilai skala dari model proses bisnis A dan B

3.2.3 Simulasi Pertumbuhan

Dalam mensimulasikan proses bisnis terdapat syarat yang harus dipenuhi yaitu model proses bisnis yang akan disimulasi harus *scalable*. Berikut algoritma yang diusulkan dengan menentukan beberapa aturan untuk simulasi pertumbuhan model proses bisnis yang dapat dilihat seperti di bawah ini:



Gambar 3.22 *Proposed Algorithm*

Pada algoritma yang dijelaskan pada **Gambar 3.22** di atas bahwa simulasi pertumbuhan pada model proses bisnis yang *scalable* akan ditumbuhkan dengan teori *Production Rule Cellular Automata* (Hopcroft, et al., 2006).

Ada beberapa aturan produksi antara lain yang digunakan dalam teori *Production Rule Cellular Automata*:

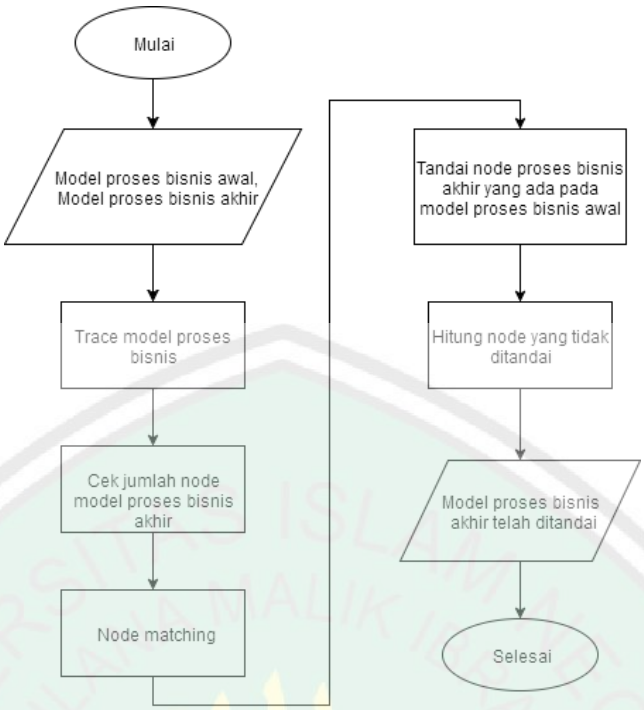
1. Aturan produksi $\alpha \rightarrow \beta$ yang diterapkan pada suatu string $w = a\alpha c$ mengganti kemunculan α menjadi β sehingga string tersebut menjadi $w = a\beta c$ sehingga daat ditulis $a\alpha c \Rightarrow a\beta c$ $(a\alpha c\ memproduksi\ a\beta c)$.
2. Produksi tersebut dapat diterapkan berkali-kali. Persamaan di bawah ini merujuk pada persamaan 2.12 seperti:

$$w_1 \Rightarrow w_2 \Rightarrow w_3 \Rightarrow w_n$$

3. Atau dapat ditulis $w_1 \Rightarrow^* w_n$, jika minimal harus ada 1 aturan produksi yang ditetapkan : $w_1 \Rightarrow^+ w_n$

3.2.4 *Mapping* Model Proses Bisnis

Mapping model proses bisnis sebelum dan sesudah disimulasikan pertumbuhannya diperlukan untuk mengetahui elemen mana saja yang baru sehingga dapat dibandingkan dengan model proses bisnis yang awal sebelum disimulasi. Maka peneliti menggunakan algoritma usulan untuk memetakan agar proses bisnis tidak saling tumpang tindih dan dapat dilihat perbedaannya sebelum dan sesudah disimulasi. Pemetaan dapat dilakukan dengan algoritma yang diusulkan seperti **Gambar 3.23** berikut ini:



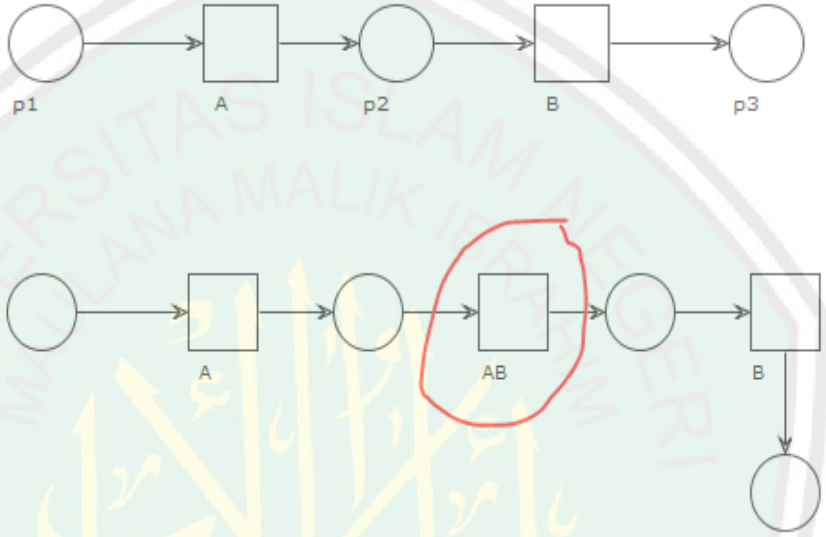
Gambar 3.23 Algoritma *Mapping* Model Proses Bisnis

Berdasarkan **Gambar 3.23** di atas, langkah *mapping* proses bisnis sebagai berikut:

1. Parameter yang digunakan adalah model proses bisnis awal dan model proses bisnis akhir.
2. *Trace* model proses bisnis
3. Selanjutnya cek berapa jumlah dari node model proses bisnis setelah disimulasi.
4. *Node Matching*
5. Menandai *node* pada model proses bisnis akhir yang terdapat pada model proses bisnis awal
6. Menghitung *node* yang tidak ditandai
7. *Node* pada model proses bisnis akhir telah ditandai

3.2.5 Penamaan Model Proses Bisnis dari Simulasi Pertumbuhan

Penamaan pada model proses bisnis yang dimaksudkan adalah penamaan elemen baru pada model proses bisnis dari hasil simulasi pertumbuhan. Contoh penamaan elemen baru pada model proses bisnis dapat dilihat pada **Gambar 3.24** di bawah ini:



Gambar 3.24 Contoh Penamaan Elemen Baru

Pada contoh **Gambar 3.24** di atas, dijelaskan terdapat model proses bisnis sebelum dan sesudah ditumbuhkan. Pada model proses bisnis yang sudah ditumbuhkan terdapat tambahan elemen baru hasil dari simulasi, sehingga peneliti mengusulkan beberapa aturan penamaan elemen baru. Misalnya seperti contoh gambar di atas elemen baru atau transisi baru terdapat pada transisi A dan B sehingga penamaan menjadi AB. Penamaan elemen pada model proses bisnis akhir dilakukan secara manual.

3.3 Pengujian Sistem

Pengujian sistem yang telah dibuat kemudian diuji untuk mengetahui kelayakan sistem. Pengujian dilakukan dengan menjadikan *input*-an dari empat

jenis model proses bisnis ERP Pondok Pesantren yang dimodelkan dengan *Petri net*. Langkah pertama yang dilakukan dengan mengukur *scalability* yang melibatkan beberapa elemen yaitu skala dan *similarity* dari suatu *workflow* (Ainul, et al., 2016). Selanjutnya, setelah dihitung nilai *scalability*-nya, model proses bisnis dapat ditumbuhkan dengan *Production Rule Cellular Automata* sehingga didapatkan beberapa variasi proses bisnis berdasarkan kemungkinannya (Wolfram, 1983). Kemudian dilakukan pemetaan proses bisnis sebelum dan sesudah disimulasikan. Terakhir, penamaan proses bisnis dari simulasi merupakan tahap terakhir. *Output* dari simulasi ini berupa file *Petri net* yang telah ditumbuhkan proses bisnisnya.

3.4 Kebutuhan Fungsional Sistem

Kebutuhan fungsional sistem adalah kebutuhan utama sistem yang di dalamnya berisi proses-proses yang nantinya akan dilakukan oleh sistem. Kebutuhan fungsional sendiri adalah layanan sistem yang harus disediakan, bagaimana sistem bereaksi terhadap *input*-an tertentu dan bagaimana perilaku sistem pada keadaan tertentu. Daftar kebutuhan fungsional sistem ditunjukkan pada **Tabel 3.1** seperti di bawah ini:

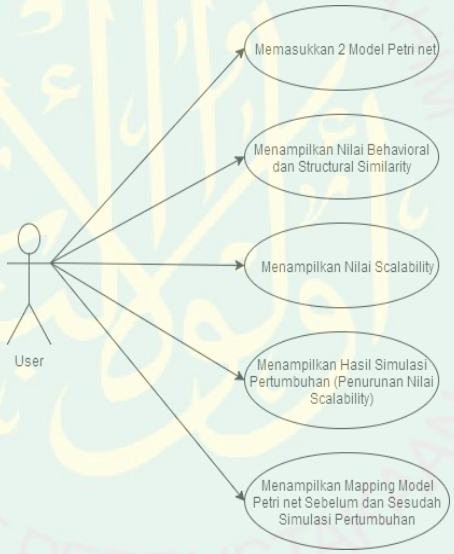
Tabel 3.1 Daftar Kebutuhan Fungsional Sistem

No	Daftar Kebutuhan Fungsional Sistem	Deskripsi
1.	*Input*-an data model proses bisnis dengan *Petri net*.	Meng-*input* model proses bisnis dengan *Petri net* yang nantinya dihitung kemiripan *workflow*, *scalability* dan simulasi pertumbuhannya.
2.	Menampilkan hasil perhitungan *similarity behavioral*, *similarity structural* dan *intersection* dari kedua model secara *structural* dan *behavioral*.	*User* dapat melihat hasil dari perhitungan *similarity behavioral*, *similarity structural* serta *intersection* dari model proses bisnis.
3.	Menampilkan hasil perhitungan *scalability*	*User* dapat melihat hasil dari perhitungan *scalability* dari model proses bisnis.
4.	Menampilkan hasil simulasi pertumbuhan proses bisnis.	*User* dapat melihat hasil dari simulasi pertumbuhan proses bisnis yang

No	Daftar Kebutuhan Fungsional Sistem	Deskripsi
		mengacu pada perubahan nilai *scalability* yakni
5.	Menampilkan *mapping* model proses bisnis dengan bertumbuhnya *node* baru pada proses bisnis	*User* dapat melihat hasil dari *mapping* yaitu dengan menampilkan *node* baru hasil dari simulasi pertumbuhan proses bisnis.
6.	Men-*regenerate* hasil simulasi pertumbuhan proses bisnis ke bentuk *file Petri net*.	*User* dapat men-*regenerate* hasil simulasi pertumbuhan proses bisnis ke bentuk *file Petri net*.

3.4.1 Analisa Penggunaan Sistem

Analisa penggunaan sistem adalah detail bagaimana *user* menggunakan sistem serta spesifikasi penggunaan terhadap sistem. Detail penggunaan sistem ditunjukkan pada **Gambar 3.25** di bawah ini:



Gambar 3.25 Diagram Penggunaan Sistem

Berdasarkan **Gambar 3.25** di atas, berikut di bawah ini akan dijelaskan analisa penggunaan user terhadap sistem, yaitu:

1. Memasukkan Dua Model PNML

   Diagram penggunaan ini menjelaskan bahwa sistem menerima *input*-an dari *user* berupa dua model PNML. Dua model PNML dimasukkan secara berurutan

yaitu model PNML yang pertama sebagai *input*-an A dan model PNML yang kedua sebagai *input*-an B.

2. Menampilkan Nilai *Behavioral* dan *Structural Similarity*

   Pada diagram penggunaan ini, sistem akan melakukan perhitungan *behavioral* dan *structural similarity*. Selain itu, ditampilkan pula *intersect* dari kedua model PNML yang di-*input*-kan *user*.

3. Menampilkan Nilai *Scalability*

   Pada diagram penggunaan ini, sistem akan melakukan perhitungan *scalability*.

4. Menampilkan Hasil Simulasi Pertumbuhan

   Pada diagram penggunaan ini, sistem akan melakukan simulasi pertumbuhan model PNML. Model PNML yang akan ditumbuhkan adalah *input*-an A yang dibandingkan dengan *input*-an B.

5. Menampilkan *Mapping* Model PNML atau *Petri net*

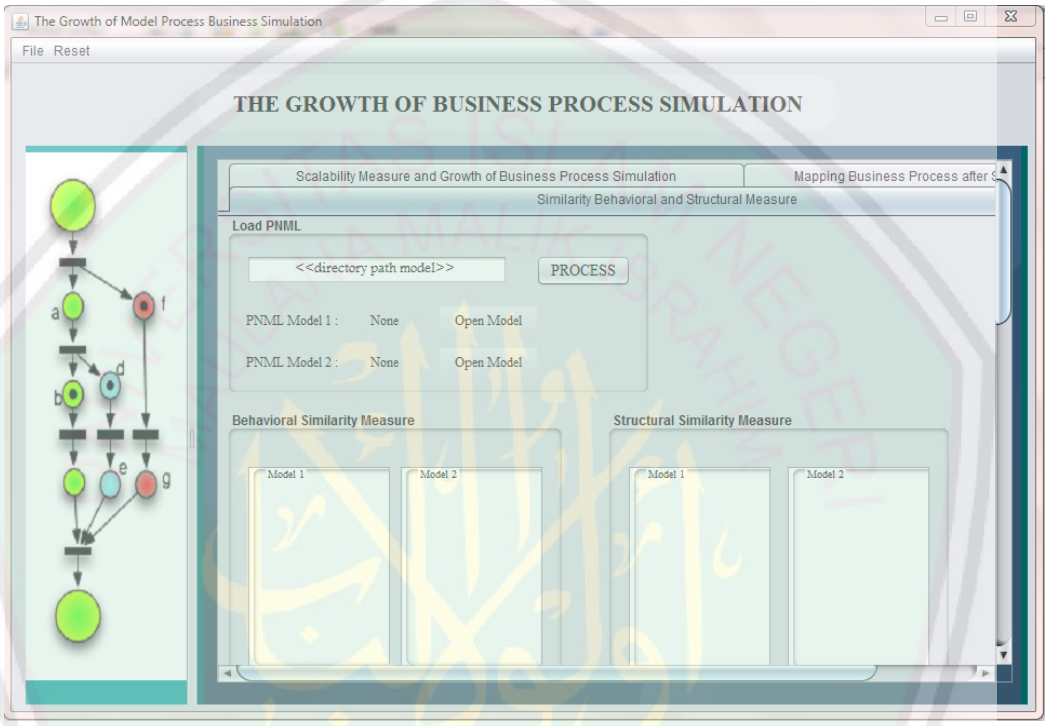
   Pada diagram penggunaan ini, sistem akan melakukan *mapping* terhadap model PNML. Model PNML yang akan dibandingkan adalah *input*-an A terhadap model PNML hasil dari simulasi pertumbuhan. Sehingga dapat diketahui *node* baru setelah simulasi pertumbuhan.

3.5 Perancangan *Interface*

Perancangan *interface* membahas bagaimana rancangan tampilan sistem atau *interface* aplikasi yang dibangun. Terdapat beberapa rancangan *interface* dari aplikasi yang dibangun diantaranya:

3.5.1 *Interface Home* Aplikasi

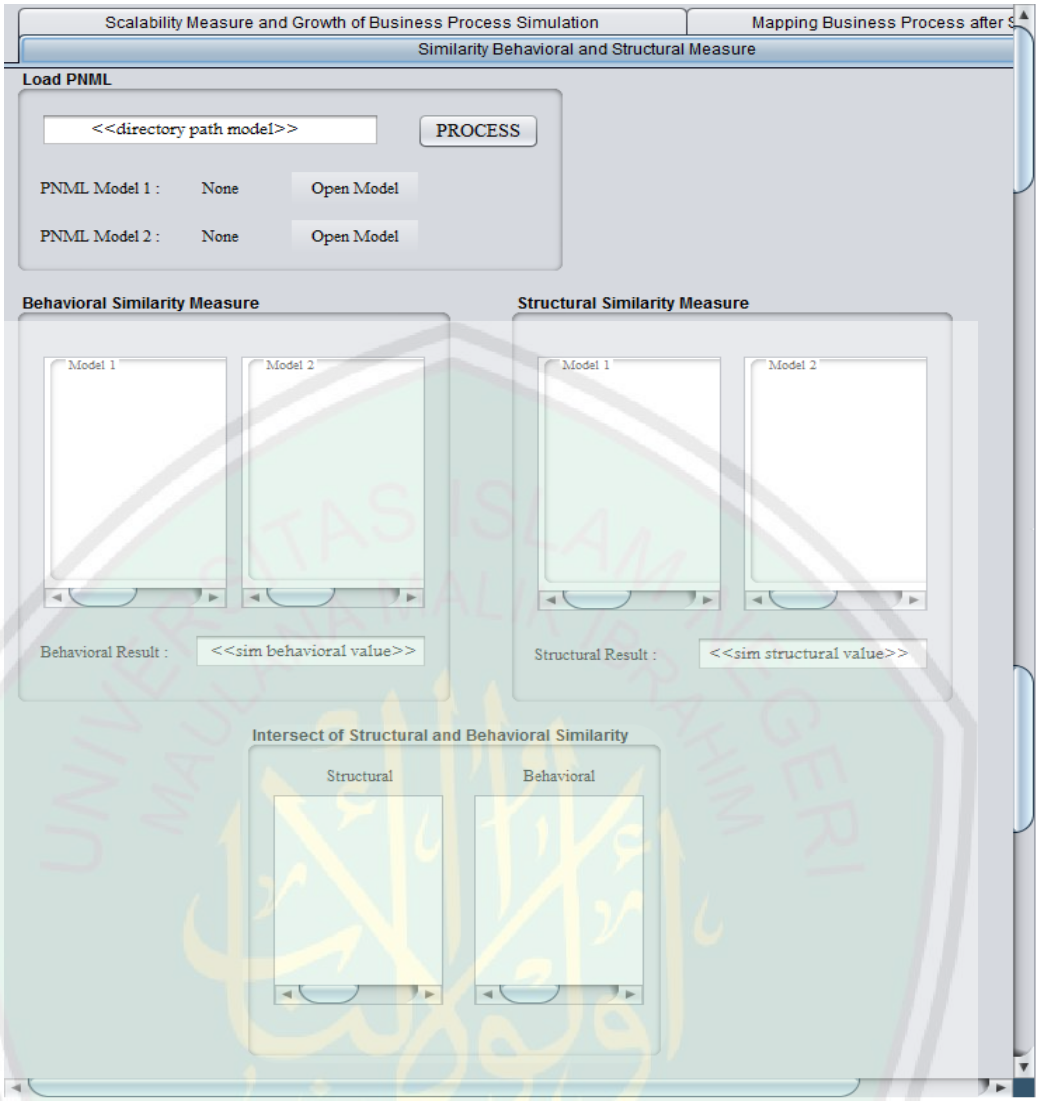
Interface Home Aplikasi merupakan antar muka halaman utama aplikasi yang dibangun. Terdapat beberapa fasilitas di halaman utama yaitu adanya menu bar yang memuat beberapa perhitungan yang ada di dalam sistem. *Interface home* ditunjukkan pada **Gambar 3.26** seperti di bawah ini:



Gambar 3.26 *Interface Home*

3.5.2 *Interface Similarity Measure*

Interface Similarity Measure digunakan sebagai *interface* untuk perhitungan *structural* dan *behavioral similarity* yang ada di dalam sistem. Berdasarkan **Gambar 3.27** di bawah ini menunjukkan *interface similirity* sistem sebagai berikut:



Gambar 3.27 *Interface Similarity Measure*

Fungsi utama dari *interface similarity* pada **Gambar 3.27** di atas adalah untuk menghitung kemiripan dari model proses bisnis yang berbentuk model PNML. Berikut atribut yang ada di dalam *interface similarity* ditunjukkan pada **Tabel 3.2** di bawah:

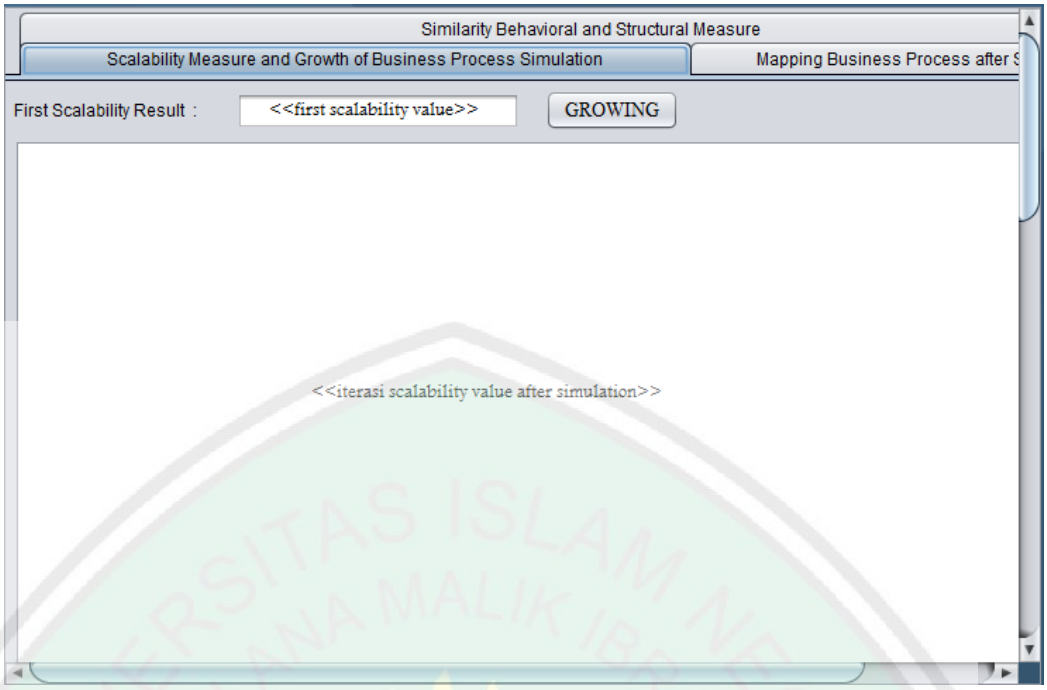
Tabel 3.2 Atribut *Interface Similarity Mesaure*

No	Nama Atribut	Jenis Atribut	Fungsi	Jenis *Input* / Output
1.	*Directory path model*	textBox	Menampilkan *directory path* model 1 dan 2	String
2.	Tombol PROCESS	actionButton	Melakukan pemrosesan model PNML	Action

3.	Tombol Open Model	actionButton	Mencari *directory path* model 1 dan 2	Action
4.	TextArea Behavioral Model 1	textBox	Menampilkan elemen secara *behavioral* model 1	String
5	TextArea Behavioral Model 2	textBox	Menampilkan elemen secara *behavioral* model 2	String
6.	TextArea Structural Model 1	textBox	Menampilkan elemen secara struktural model 1	String
7.	TextArea Structural Model 2	textBox	Menampilkan elemen secara struktural model 2	String
8.	Sim value behavioral	textBox	Menampilkan nilai kemiripan secara *behavioral*	Double
9.	Sim value structural	textBox	Menampilkan nilai kemiripan secara struktural	Double
10.	TextArea Intersect Structural	textBox	Menampilkan intersect secara struktural	String
11	TextArea Intersect Behavioral	textBox	Menampilkan intersect secara *behavioral*	String
12.	None	Text	Menampilkan nama *file* PNML	String

3.5.3 *Interface Scalability and Growth of Business Prosess Simulation*

Interface Scalability dan *Growth of Business Process* Simulation digunakan sebagai *interface* untuk menghitung nilai *scalability* dan simulasi pertumbuhan proses bisnis. Berikut *interface scalability* dan simulasi pertumbuhan ditunjukkan pada **Gambar 3.28** di bawah ini:



Gambar 3.28 *Interface Scalability* dan *The Growth Business Process Simulation*

Fungsi utama dari *interface similarity* pada **Gambar 3.28** di atas adalah untuk menghitung nilai *scalability* dan simulasi pertumbuhan proses bisnis berdasarkan nilai *scalability* yang menurun. Berikut atribut yang ada di dalam *interface scalability* dan simulasi pertumbuhan ditunjukkan pada **Tabel 3.3** di bawah ini:

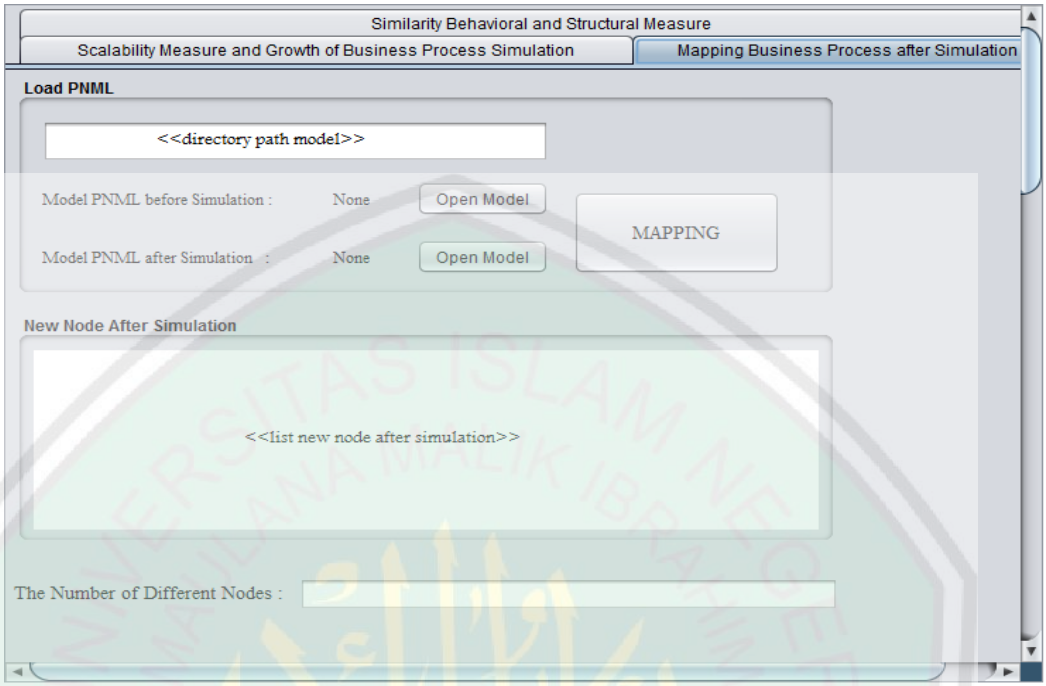
Tabel 3.3 Atribut *Interface Scalability* dan *Simulation*

No	Nama Atribut	Jenis Atribut	Fungsi	Jenis *Input* / Output
1.	*First scalability value*	textBox	Menmpilkan nilai *scalability* awal sebelum disimulasi pertumbuhan model PNML	String
2.	Tombol GROWING	actionButton	Melakukan pemrosesan model PNML untuk simulasi dan *generate* ke file PNML	Action
3.	TextArea	textBox	Menampilkan iterasi nilai *scalability*	String

3.5.4 *Interface Mapping Business Process*

Interface Mapping Business Process digunakan sebagai antarmuka perbandingan model PNML awal dengan model PNML setelah simulasi,

kemudian dapat diketahui *node* atau elemen baru. Berikut *interface mapping* ditunjukkan pada **Gambar 3.29** di bawah ini:



Gambar 3.29 *Interface Mapping Business Process*

Fungsi utama dari *interface similarity* pada **Gambar 3.29** di atas adalah untuk membandingkan model PNML awal dengan model PNML setelah simulasi. Berikut atribut yang ada di dalam *interface mapping* ditunjukkan pada **Tabel 3.4** di bawah ini:

Tabel 3.4 Atribut *Interface Mapping* Proses Bisnis

No	Nama Atribut	Jenis Atribut	Fungsi	Jenis *Input* / Output
1.	Field *Directory path model*	textBox	Menmpilkan nilai *scalability* awal sebelum disimulasi pertumbuhan model PNML	String
2.	None	Text	Menampilkan nama *file* PNML	String
3.	Tombol Open Model	actionButton	Mencari *directory path* model 1 dan 2	Action
4.	Tombol MAPPING	actionButton	Melakukan pemroresan objek file PNML	Action
5.	TextArea node Baru	textBox	Menampilkan node baru setelah dilakukan simulasi pertumbuhan proses bisnis	String

No	Nama Atribut	Jenis Atribut	Fungsi	Jenis *Input* / Output
6.	Field jumlah *node* yang berbeda	textBox	Menampilkan jumlah node yang berbeda	Int

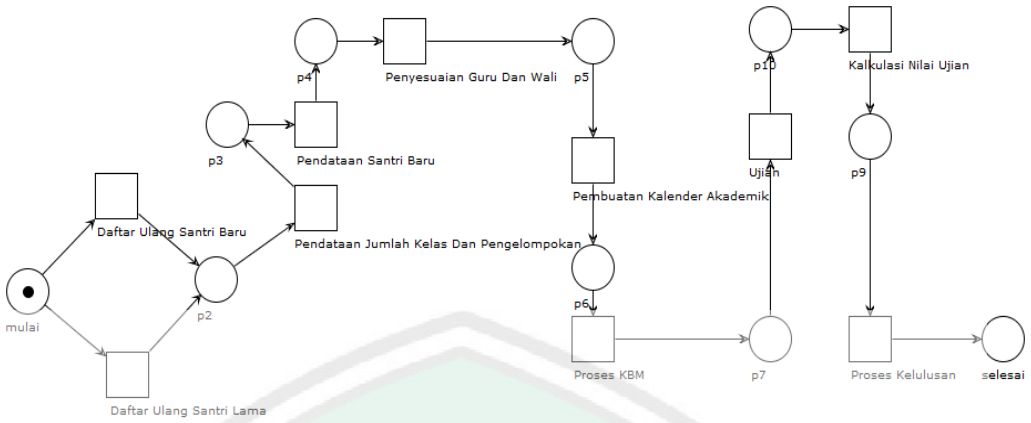
BAB 4

HASIL DAN PEMBAHASAN

Bab ini menjelaskan hasil dan pembahasan sistem dari yang telah dibangun. Banyaknya analisa dari bab sebelumnya, maka pada bab ini dilakukan pembahasan terhadap aplikasi Simulasi Pertumbuhan *Scalable Business Process Model*. Oleh karena itu, dilakukan pengujian terhadap sistem diantaranya mengolah data PNML dari empat tipe pondok dengan lima bagian utama dari masing-masing tipe sebagai data uji. Kemudian melakukan *parsing*, menghitung nilai *structural* dan *behavioral similarity*, menghitung *complexity* elemen model, menghitung *scalability* dan menumbuhkan proses bisnis dengan acuan iterasi penurunan nilai *scalability*.

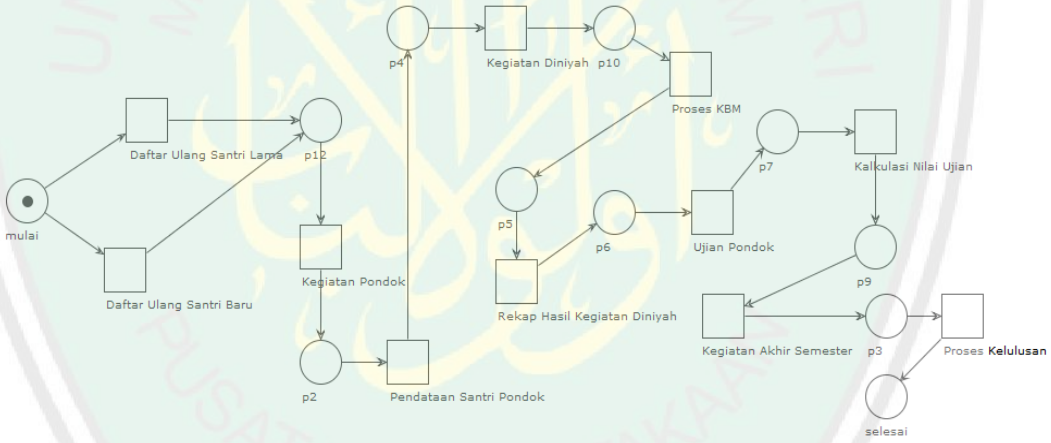
4.1 Model PNML Sebagai Data Uji

Model PNML yang digunakan sebagai data uji adalah model yang dibangun menggunakan *Petri net*. Model tersebut merupakan proses bisnis dari empat tipe Pondok Pesantren yakni: Mahasiswa, *Modern*, *Salaf* dan *Tahfidz*. Data uji tersebut digunakan sebagai data masukan untuk dilakukan *parsing*. Data masukan yang digunakan adalah dua data *input*-an dari tipe pondok berbeda namun dari bagian yang sama. Data uji yang digunakan terdapat pada **Gambar 3.2, Gambar 3.3, Gambar 3.4, Gambar 3.5, Gambar 3.6, Gambar 3.7, Gambar 3.8, Gambar 3.9, Gambar 3.10, Gambar 3.11, Gambar 3.12, Gambar 3.13, Gambar 3.14, Gambar 3.15, Gambar 3.16, Gambar 3.17, Gambar 3.18, Gambar 3.19, Gambar 3.20** dan **Gambar 3.21** di bab sebelumnya. Contoh model PNML yang digunakan sebagai masukan ditunjukkan pada **Gambar 4.1** di bawah ini:



Gambar 4.1 Proses Bisnis Akademik *Modern*

Pada **Gambar 4.1** di atas merupakan data *input*-an A. *Input*-an A merupakan proses model yang lebih sederhana artinya jumlah *node* dan *complexity* lebih kecil dibandingkan dengan *input*-an B.



Gambar 4.2 Proses Bisnis Akademik Mahasiswa

Pada **Gambar 4.2** di atas merupakan data *input*-an B. *Input*-an B merupakan proses model yang lebih komplek artinya jumlah *node* dan *complexity* lebih besar dibandingkan dengan *input*-an A.

Input-an A nantinya akan ditumbuhkan proses bisnisnya, dengan *input*-an B sebagai pembanding perhitungan iterasi penurunan nilai *scalability*.

4.2 Parsing Model PNML

Parsing model PNML adalah membaca dan mengambil nilai dari *node* atau elemen dari setiap model yang di-*input*-kan. Terdapat tiga elemen atau yang disebut sebagai *node* dalam *Petri net* yakni: *place*, *transisi* dan *arc*. Kegunaan *parsing* yaitu dapat menghitung nilai *structural* dan *behavioral similarity workflow*, menghitung nilai *scalability*, proses simulasi pertumbuhan proses bisnis dan *mapping* proses bisnis setelah simulasi. Pada dasarnya model PNML atau model *Petri net* adalah *file* yang berbasis *xml*, oleh karena itu jenis Java DOM Parser dirasa cocok untuk digunakan dalam proses *parsing* model PNML. Atribut-atribut dari model PNML berbasis *xml* ditunjukkan pada **Kode sumber 4.1, Kode sumber 4.2, Kode sumber 4.3** seperti di bawah ini:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--PLEASE DO NOT EDIT THIS FILE
Created with Workflow PetriNet Designer Version 3.2.0 (woped.org)-->
<pnml>
  <net type="http://www.informatik.hu-berlin.de/top/pntd/ptNetb"
id="noID">
    <place id="p1">
      <name>
        <text>mulai</text>
        <graphics>
          <offset x="40" y="210"/>
        </graphics>
      </name>
      <graphics>
        <position x="40" y="170"/>
        <dimension x="40" y="40"/>
      </graphics>
      <initialMarking>
        <text>1</text>
      </initialMarking>
    </place>
```

Kode sumber 4.1 Atribut *Node Place* Proses Bisnis Akademik Mahasiswa

```
<transition id="t4">
      <name>
        <text>Kegiatan Akhir Semester</text>
        <graphics>
          <offset x="680" y="310"/>
        </graphics>
      </name>
      <graphics>
        <position x="680" y="270"/>
        <dimension x="40" y="40"/>
```

```
      </graphics>
      <toolspecific tool="WoPeD" version="1.0">
        <time>0</time>
        <timeUnit>1</timeUnit>
        <orientation>1</orientation>
      </toolspecific>
    </transition>
```

Kode sumber 4.2 Atribut *Node Transition* Proses Bisnis Akademik Mahasiswa

```
<arc id="a11" source="p6" target="t7">
      <inscription>
        <text>1</text>
      </inscription>
      <graphics/>
      <toolspecific tool="WoPeD" version="1.0">
        <probability>1.0</probability>
        <displayProbabilityOn>false</displayProbabilityOn>
        <displayProbabilityPosition x="500.0" y="0.0"/>
      </toolspecific>
    </arc>
```

Kode sumber 4.3 Atribut *Node Arc* Proses Bisnis Akademik Mahasiswa

Atribut *node* dalam *file* xml tersebut kemudian dibaca oleh Java DOM Parser dengan normalisasi *file* xml. Normalisasi *file* xml dengan menempatkan *file* di dalam direktori ke dalam *Class File*, kemudian *file* xml tersebut di-*parsing* ke dalam bentuk *Document* dengan *class* `DocumentBuilder`. *Document* tersebut berfungsi untuk menampung sintaks, elemen dan atribut xml agar dapat terbaca oleh Java DOM Parser dalam proses normalisasi. Namun, *value* dari *file* PNML tersebut belum terambil. Berikut proses normalisasi model PNML dalam bentuk *file* xml ditunjukkan pada **Kode sumber 4.4** seperti di bawah ini:

```
File inputFile = new File(model);
DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
Document doc = dBuilder.parse(inputFile);
doc.getDocumentElement().normalize();
```

Kode sumber 4.4 *Parsing* model PNML

Jika proses normalisasi selesai, kemudian memanggil beberapa elemen xml dengan *TagName* "*arc*", "*place*" dan "*transition*" yang ditampung dalam objek `NodeList`. Berikut ditunjukkan pada **Kode sumber 4.5** di bawah ini:

```
NodeList narc = doc.getElementsByTagName("arc");
NodeList nplace = doc.getElementsByTagName("place");
NodeList ntransisi = doc.getElementsByTagName("transition");
```

Kode sumber 4.5 Memanggil Elemen XML dengan *TagName*

Selanjutnya, pengambilan nilai berdasarkan *TagName* yang telah didefinisikan, berdasarkan nilai atribut yang terdapat dalam *file* xml, jumlah elemen dalam model PNML tersebut dapat dihitung. Berikut adalah pengambilan nilai dari elemen *place* serta jumlah elemennya ditunjukkan pada **Kode sumber 4.6** seperti di bawah ini:

```
NodeList nplace = doc.getElementsByTagName("place");
for (int i = 0; i < nplace.getLength(); i++) {
plc1 =
nplace.item(i).getAttributes().getNamedItem("id").getNodeValue();
                System.out.println("" + plc1);
                jumlahNode1++;
                if (Place1[0] == null) {
                    Place1[0] = plc1;
                } else if (Place1[1] == null) {
                    Place1[1] = plc1;
                } else if (Place1[2] == null) {
                    Place1[2] = plc1;
                } else if (Place1[3] == null) {
                    Place1[3] = plc1;
                } else if (Place1[4] == null) {
                    Place1[4] = plc1;
                } else if (Place1[5] == null) {
                    Place1[5] = plc1;
                } else if (Place1[6] == null) {
                    Place1[6] = plc1;
                } else if (Place1[7] == null) {
                    Place1[7] = plc1;
                } else if (Place1[8] == null) {
                    Place1[8] = plc1;
                } else if (Place1[9] == null) {
                    Place1[9] = plc1;
                } else if (Place1[10] == null) {
                    Place1[10] = plc1;
                } else if (Place1[11] == null) {
                    Place1[11] = plc1;
                } else if (Place1[12] == null) {
                    Place1[12] = plc1;
                } else if (Place1[13] == null) {
                    Place1[13] = plc1;
                } else if (Place1[14] == null) {
                    Place1[14] = plc1;
                } else if (Place1[15] == null) {
                    Place1[15] = plc1;
                } else if (Place1[16] == null) {
                    Place1[16] = plc1;
                } else if (Place1[17] == null) {
                    Place1[17] = plc1;
                } else if (Place1[18] == null) {
                    Place1[18] = plc1;
```

```
                } else if (Place1[19] == null) {
                    Place1[19] = plc1;
                }

            }
```

Kode sumber 4.6 Pemanggilan dan Menghitung Jumlah Elemen *Place*

Setelah **Kode sumber 4.6** di atas dijalankan, menghasilkan *output* hasil *parsing* berdasarkan *value* "id" dari elemen *place*. Keluaran yang dihasilkan berupa elemen *place* dari atribut pada model PNML yang telah di*input*kan. Berikut *output* ditunjukkan seperti pada **Gambar 4.3** di bawah ini:

```
run:
p1
p2
p3
p4
p5
p6
p7
p8
p9
p10
----------
```

Gambar 4.3 Hasil *Parsing* Proses Bisnis Akademik Modern Elemen *Place*

Pengambilan nilai elemen transisi pada *file* xml serta menghitung jumlah elemen transisi di dalamnya ditunjukkan pada **Kode sumber 4.7** seperti di bawah ini:

```
NodeList ntransisi = doc.getElementsByTagName("transition");
for (int i = 0; i < ntransisi.getLength(); i++) {
trans1 =
ntransisi.item(i).getAttributes().getNamedItem("id").getNodeValue();
                System.out.println("" + trans1);
                jumlahNode1++;
                if (Transisi1[0] == null) {
                    Transisi1[0] = trans1;
                } else if (Transisi1[1] == null) {
                    Transisi1[1] = trans1;
                } else if (Transisi1[2] == null) {
                    Transisi1[2] = trans1;
                } else if (Transisi1[3] == null) {
                    Transisi1[3] = trans1;
                } else if (Transisi1[4] == null) {
                    Transisi1[4] = trans1;
                } else if (Transisi1[5] == null) {
                    Transisi1[5] = trans1;
                } else if (Transisi1[6] == null) {
                    Transisi1[6] = trans1;
```

```
            } else if (Transisi1[7] == null) {
                Transisi1[7] = trans1;
            } else if (Transisi1[8] == null) {
                Transisi1[8] = trans1;
            } else if (Transisi1[9] == null) {
                Transisi1[9] = trans1;
            } else if (Transisi1[10] == null) {
                Transisi1[10] = trans1;
            } else if (Transisi1[11] == null) {
                Transisi1[11] = trans1;
            } else if (Transisi1[12] == null) {
                Transisi1[12] = trans1;
            } else if (Transisi1[3] == null) {
                Transisi1[13] = trans1;
            } else if (Transisi1[14] == null) {
                Transisi1[14] = trans1;
            } else if (Transisi1[5] == null) {
                Transisi1[15] = trans1;
            } else if (Transisi1[16] == null) {
                Transisi1[16] = trans1;
            } else if (Transisi1[17] == null) {
                Transisi1[17] = trans1;
            } else if (Transisi1[18] == null) {
                Transisi1[18] = trans1;
            } else if (Transisi1[19] == null) {
                Transisi1[19] = trans1;
            }
        }
```

Kode sumber 4.7 Pemanggilan dan Menghitung Jumlah Elemen Transisi

Setelah **Kode sumber 4.7** di atas dijalankan, menghasilkan *output* hasil *parsing* berdasarkan *value* "id" dari elemen transisi. Elemen yang dihasilkan berupa *node* transisi dari atribut pada model PNML. Berikut *output* yang ditunjukkan seperti **Gambar 4.4** di bawah ini:

```
t5
t6
t7
t8
t9
t14
t15
t1
t2
t3
----------
```

Gambar 4.4 Hasil *Parsing* Proses Bisnis Akademik Modern Elemen *Transisi*

Pengambilan nilai elemen *arc* pada *file* xml dan jumlah elemen di dalam model PNML. Elemen *arc* terdiri dari "*source*" dan "target". "*Source*" menunjukkan asal

busur *arc* sedangkan “target” menunjukkan tujuan busur *arc* .Ditunjukkan pada **Kode sumber 4.8** seperti di bawah ini:

```
for (int i = 0; i < narc.getLength(); i++) {
String arc1 =
narc.item(i).getAttributes().getNamedItem("source").getNodeValue();
String arc2 =
narc.item(i).getAttributes().getNamedItem("target").getNodeValue();
                String unitarc = arc1 + "->" + arc2;
                System.out.println("" + unitarc);
                jumlahNode1++;
                if (Arc1[0] == null) {
                    Arc1[0] = unitarc;
                } else if (Arc1[1] == null) {
                    Arc1[1] = unitarc;
                } else if (Arc1[2] == null) {
                    Arc1[2] = unitarc;
                } else if (Arc1[3] == null) {
                    Arc1[3] = unitarc;
                } else if (Arc1[4] == null) {
                    Arc1[4] = unitarc;
                } else if (Arc1[5] == null) {
                    Arc1[5] = unitarc;
                } else if (Arc1[6] == null) {
                    Arc1[6] = unitarc;
                } else if (Arc1[7] == null) {
                    Arc1[7] = unitarc;
                } else if (Arc1[8] == null) {
                    Arc1[8] = unitarc;
                } else if (Arc1[9] == null) {
                    Arc1[9] = unitarc;
                } else if (Arc1[10] == null) {
                    Arc1[10] = unitarc;
                } else if (Arc1[11] == null) {
                    Arc1[11] = unitarc;
                } else if (Arc1[12] == null) {
                    Arc1[12] = unitarc;
                } else if (Arc1[13] == null) {
                    Arc1[13] = unitarc;
                } else if (Arc1[14] == null) {
                    Arc1[14] = unitarc;
                } else if (Arc1[15] == null) {
                    Arc1[15] = unitarc;
                } else if (Arc1[16] == null) {
                    Arc1[16] = unitarc;
                } else if (Arc1[17] == null) {
                    Arc1[17] = unitarc;
                } else if (Arc1[18] == null) {
                    Arc1[18] = unitarc;
                } else if (Arc1[19] == null) {
                    Arc1[19] = unitarc;
                } else if (Arc1[20] == null) {
                    Arc1[20] = unitarc;
                } else if (Arc1[21] == null) {
                    Arc1[21] = unitarc;
                } else if (Arc1[22] == null) {
                    Arc1[22] = unitarc;
                } else if (Arc1[23] == null) {
                    Arc1[23] = unitarc;
                } else if (Arc1[24] == null) {
                    Arc1[24] = unitarc;
                } else if (Arc1[25] == null) {
```

```
                Arc1[25] = unitarc;
            } else if (Arc1[26] == null) {
                Arc1[26] = unitarc;
            } else if (Arc1[27] == null) {
                Arc1[27] = unitarc;
            } else if (Arc1[28] == null) {
                Arc1[28] = unitarc;
            } else if (Arc1[29] == null) {
                Arc1[29] = unitarc;
            } else if (Arc1[30] == null) {
                Arc1[30] = unitarc;
            } else if (Arc1[31] == null) {
                Arc1[31] = unitarc;
            } else if (Arc1[32] == null) {
                Arc1[32] = unitarc;
            } else if (Arc1[33] == null) {
                Arc1[33] = unitarc;
            } else if (Arc1[34] == null) {
                Arc1[34] = unitarc;
            } else if (Arc1[35] == null) {
                Arc1[35] = unitarc;
            } else if (Arc1[36] == null) {
                Arc1[36] = unitarc;
            } else if (Arc1[37] == null) {
                Arc1[37] = unitarc;
            } else if (Arc1[38] == null) {
                Arc1[38] = unitarc;
            } else if (Arc1[39] == null) {
                Arc1[39] = unitarc;
            }
        }
```

Kode sumber 4.8 Pemanggilan dan Menghitung Jumlah Elemen *Arc*

Setelah **Kode sumber 4.8** di atas dijalankan, menghasilkan *output* hasil parsing berdasarkan *value* "id" dari elemen *arc* yang ditunjukkan pada **Gambar 4.5** seperti di bawah ini:

```
t5->p6
t9->p10
p5->t5
p9->t9
t7->p8
p6->t6
p7->t7
t6->p7
t8->p9
p8->t8
p1->t2
p1->t1
t1->p2
p2->t14
t14->p3
p3->t3
t3->p4
p4->t15
t15->p5
t2->p2
```

Gambar 4.5 Hasil *Parsing* Proses Bisnis Akademik Modern Elemen *Arc*

Berdasarkan **Gambar 4.5** di atas menunjukkan "t5-p6" artinya "*source*" *arc* berada di elemen "t5" dan "*target*" *arc* berada pada elemen "p6".

Setelah masing-masing elemen dihitung, kemudian dihitung jumlah akumulasi tiga elemen pada model PNML atau *Petri net*. Seperti yang sudah dijelaskan di atas terdapat dua *input*-an yang di-*parsing*. Pada percobaan *parsing* di atas menggunakan Proses Bisnis Akademik *Modern* sebagai *input*-an A dan Proses Bisnis Akademik Mahasiswa sebagai *input*-an B.

Berdasarkan **Kode sumber 4.1** dan **Kode sumber 4.2** merupakan kode sumber dari *class ParserStructur* yang menangani *parsing* secara struktural. Kode sumber yang telah dipaparkan di atas juga berlaku sama pada *class ParserBehavioal* yang menangani *parsing* secara *behavioral*. Namun ada yang membedakan dalam *parsing* secara *behavioral* yaitu pada pengambilan nilai transisi yang terpisah akan dikumpulkan menjadi bentuk TARset. *Parsing* model PNML untuk membentuk TARset ditangani oleh *method* `ParsingTAR1(String model)` yang ditunjukkan pada **Kode sumber 4.9** di bawah ini:

```
public void ParsingTAR1(String model) {
        for (int i = 0; i < set.length; i++) {
            set[i] = null;
        }

try {
   File inputFile = new File(model);
   DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
   DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
   Document doc = dBuilder.parse(inputFile);
  doc.getDocumentElement().normalize();

  NodeList narc = doc.getElementsByTagName("arc");
  NodeList nplace = doc.getElementsByTagName("place");
  NodeList ntransisi = doc.getElementsByTagName("transition");

 for (int i = 0; i < narc.getLength(); i++) {
      arc11 = narc.item(i).getAttributes().getNamedItem
("target").getNodeValue();
      trans1 = narc.item(i).getAttributes().getNamedItem
("source").getNodeValue();
      String t1 = arc11.substring(0, 1);
          for (int j = 0; j < narc.getLength(); j++) {
```

```
          arc22 = narc.item(j).getAttributes().
getNamedItem("source").getNodeValue();
           trans2 = narc.item(j).getAttributes().
getNamedItem("target").getNodeValue();
          if (arc11.equals(arc22) && !(t1.equals("t"))) {
              jumlahTaR++;
              unit = trans1 + trans2;
              System.out.println(unit);
              if (Arc1[0] == null) {
                  Arc1[0] = unit;
              } else if (Arc1[1] == null) {
                  Arc1[1] = unit;
              } else if (Arc1[2] == null) {
                  Arc1[2] = unit;
              } else if (Arc1[3] == null) {
                  Arc1[3] = unit;
              } else if (Arc1[4] == null) {
                  Arc1[4] = unit;
              } else if (Arc1[5] == null) {
                  Arc1[5] = unit;
              } else if (Arc1[6] == null) {
                  Arc1[6] = unit;
              } else if (Arc1[7] == null) {
                  Arc1[7] = unit;
              } else if (Arc1[8] == null) {
                  Arc1[8] = unit;
              } else if (Arc1[9] == null) {
                  Arc1[9] = unit;
              } else if (Arc1[10] == null) {
                  Arc1[10] = unit;
              } else if (Arc1[11] == null) {
                  Arc1[11] = unit;
              } else if (Arc1[12] == null) {
                  Arc1[12] = unit;
              } else if (Arc1[13] == null) {
                  Arc1[13] = unit;
              } else if (Arc1[14] == null) {
                  Arc1[14] = unit;
              } else if (Arc1[15] == null) {
                  Arc1[15] = unit;
              } else if (Arc1[16] == null) {
                  Arc1[16] = unit;
              } else if (Arc1[17] == null) {
                  Arc1[17] = unit;
              } else if (Arc1[18] == null) {
                  Arc1[18] = unit;
              } else if (Arc1[19] == null) {
                  Arc1[19] = unit;
              }
          }
                                  ...
```

Kode sumber 4.9 *Method* ParsingTAR ()

Berdasarkan **Kode sumber 4.9** di atas, dapat dijelaskan bahwa proses *parsing* data model PNML hingga membentuk nilai TARset, dengan *input* dari *path* direktori model tersebut.

4.3 Perhitungan *Structural* dan *Behavioral Similarity*

Perhitungan *Structural* dan *Behavioral Similarity* digunakan sebagai salah satu variabel untuk menghitung nilai *scalability*. Mencari kemiripan *workflow* pada perhitungan *structural* maupun *behavioral* menggunakan empat algoritma yaitu: *Jaccard Coefficient Similarity, Overlap Coefficient Similarity, Dice Coefficient Similarity* dan *Cosine Coefficient Similarity*.

1. *Structural Similarity*

   Perhitungan *structural similarity* yaitu menghitung kemiripan struktur dua buah *workflow*. Struktur *workflow* yang dimaksud meliputi elemen yang ada di dalam *workflow*/model proses bisnis. Terdapat beberapa metode yang digunakan untuk menghitung kemiripan struktural pada *workflow* antara lain:

   a. Metode *Jaccard Coefficient Similarity*

      Salah satu metode yang digunakan dalam perhitungan *structural similarity* yaitu *Jaccard Coefficient Similarity*. *Jaccard Coefficient* adalah perhitungan untuk membandingkan elemen *intersect* dari dua model objek yang dicari kemiripannya dengan gabungan elemen dikurangi *intersect* dari dua buah objek. Perhitungan *Jaccard Coefficient* ditangani oleh *method* `struktural(int A, int B, int Irisan)` yang ditunjukkan pada **Kode sumber 4.10** di bawah ini:

```
public double struktural(int A, int B, int Irisan) {
System.out.println("Jaccard Structural Coefficient Similarity");
        System.out.println("PetriNet 1 = " + A);
        System.out.println("PetriNet 1 = " + B);
        System.out.println("Irisan = " + Irisan);
        double atas = Irisan;
        double bawah = A + B - Irisan;
        Jc1 = atas / bawah;
        String hasil = decimalFormat.format(Jc1);
        System.out.println("" + Irisan);
        System.out.println("Jaccard Structural = " +Jc1);
        return Jc1;
    }
```

Kode sumber 4.10 Perhitungan *Jaccard Coefficient Similarity*

b. Metode *Overlap Coefficient Similarity*

Perhitungan kemiripan menggunakan metode *Overlap Coefficient Similarity* secara *structural* maksudnya adalah membandingkan antara *intersect* objek dengan nilai minimum dari objek yang dijadikan sebagai pembanding. Perhitungan metode *overlap* ditangani oleh *method* `overlap(int A, int B, int Min, int Irisan)`. Berikut ditunjukkan pada **Kode sumber 4.11** di bawah ini:

```
public double overlap(int A, int B, int Min, int Irisan) {
System.out.println("Overlap Coefficient Similarity");
        System.out.println("PetriNet 1 = " + A);
        System.out.println("PetriNet 1 = " + B);
        System.out.println("Irisan = " + Irisan);
        System.out.println("Min =" + Min);
        double atas = Irisan;
        if (A < B) {
            Min = A;
        } else {
            Min = B;
        }
        double bawah = Min;
        Ov1 = atas / bawah;
        System.out.println(" Overlap Coefficient = " + Ov1);
        return Ov1;
    }
```

Kode sumber 4.11 Perhitungan *Overlap Coefficient Similarity*

c. Metode *Dice Coefficient Similarity*

Perhitungan algoritma *Dice Coefficient Similarity* secara struktural yaitu mencari kemiripan menggunakan *intersect* yang dibandingkan dengan gabungan model objek. Hasil dari perbandingan pada operasi sebelumnya kemudian dikalikan dua. Perhitungan *Dice Coefficient* ditangani oleh *method* `dice(int A, int B, int Irisan)` yang ditunjukkan pada **Kode sumber 4.12** di bawah ini:

```
public double dice(int A, int B, int Irisan) {
```

```
System.out.println("Dice's Coefficient Similarity");
        System.out.println("Petri net 1= " + A);
        System.out.println("Petri Net 2= " + B);
        System.out.println("Irisan = " + Irisan);
        double atas = Irisan;
        double bawah = A + B;
        Dc1 = 2 * (atas / bawah);
        System.out.println("Dice's Coefficient Similarity = " +
Dc1);
        return Dc1;
    }
```

Kode sumber 4.12 Perhitungan *Dice Coefficient Similarity*

d. Metode *Cosine Coefficient Similarity*

Perhitungan algoritma *Cosine Coefficient Similarity* secara struktural adalah mencari kemiripan dengan membandingkan *intersect* objek dengan perkalian akar dari kedua objek. Perhitungan *cosine* ditangani oleh *method* `Cosine(int A, int B, int Irisan)` yang ditunjukkan pada **Kode sumber 4.13** di bawah ini:

```
public double Cosine(int A, int B, int Irisan) {
System.out.println("Cosine Coefficient Similarity");
        System.out.println("Petri Net 1= " + A);
        System.out.println("Petri Net 2= " + B);
        System.out.println("Irisan = " + Irisan);
        double atas = Irisan;
        double bawah = Math.sqrt(A) * Math.sqrt(B);
        Co1 = atas / bawah;
        System.out.println("Cosine Coefficient = " + Co1);
        return Co1;
    }
```

Kode sumber 4.13 Perhitungan *Cosine Coefficient Similarity*

2. *Behavioral Similarity*

Perhitungan *behavioral similarity* yaitu menghitung kemiripan perilaku dua buah *workflow*. Struktur *workflow* yang dimaksud meliputi elemen yang ada di dalam *workflow*/model proses bisnis. Terdapat beberapa metode yang digunakan untuk menghitung kemiripan perilaku pada *workflow* antara lain:

a. Metode *Jaccard Behavioral Similarity*

Salah satu metode yang digunakan dalam perhitungan *behavioral similarity* yaitu *Jaccard Coefficient Similarity*. *Jaccard Coefficient* adalah perhitungan untuk membandingkan elemen kuadrat *intersect* dari dua model objek yang dicari kemiripannya dengan perkalian dua buah objek model. Perhitungan *Jaccard Coefficient* ditangani oleh *method* `jaccard(int A, int B, int Irisan)` yang ditunjukkan pada **Kode sumber 4.14** di bawah ini:

```
public double jaccard(int A, int B, int Irisan) {
        System.out.println("TAR Coefficient Similarity");
        System.out.println("PetriNet 1 = " + A);
         System.out.println("PetriNet 1 = " + B);
         System.out.println("Irisan = " + Irisan);
         double atas = (Irisan) * (Irisan);
         double bawah = A * B;
         Jc2 = atas / bawah;

   System.out.println(" Jaccard Structural = " + Jc2);
        return Jc2;
}
```

Kode sumber 4.14 Perhitungan *Jaccard Coefficient Simlarity*

b. Metode *Overlap Coefficient Similarity*

Perhitungan kemiripan menggunakan metode *Overlap Coefficient Similarity* secara *behavioral* maksudnya adalah membandingkan antara *intersect* objek dengan nilai minimum dari objek yang dijadikan sebagai pembanding. Perhitungan *overlap coeffcient* secara *behavioral* sama dengan perhitungan secara struktural. Perhitungan metode *overlap* ditangani oleh *method* `overlap(int A, int B, int Min, int Irisan)`. Berikut ditunjukkan pada **Kode sumber 4.15** di bawah ini:

```
public double overlap(int A, int B, int Min, int Irisan) {
    System.out.println("Overlap Coefficient Similarity");
    System.out.println("PetriNet 1 = " + A);
    System.out.println("PetriNet 1 = " + B);
    System.out.println("Irisan = " + Irisan);
    System.out.println("Min = " + Min);
    double atas = Irisan;
```

```
    if (A < B) {
        Min = A;
       } else {
        Min = B;
       }
    double bawah = Min;
    Ov2 = atas / bawah;
    System.out.println(" Overlap Coefficient = " + Ov2);
    return Ov2;
}
```

Kode sumber 4.15 Perhitungan *Overlap Coefficient Similarity*

c. Metode *Dice Coefficient Similarity*

Perhitungan algoritma *Dice Coefficient Similarity* secara *behavioral* yaitu mencari kemiripan menggunakan *intersect* yang dibandingkan dengan gabungan model objek. Hasil dari perbandingan pada operasi sebelumnya kemudian dikalikan dua. Algoritma *dice* secara *behavioral* sama dengan perhitungan secara struktural. Perhitungan *Dice Coefficient* ditangani oleh *method* `dice(int A, int B, int Irisan)` yang ditunjukkan pada **Kode sumber 4.16** di bawah ini:

```
public double dice(int A, int B, int Irisan) {
System.out.println("Dice's Coefficient Similarity");
        System.out.println("Petri Net 1= " + A);
        System.out.println("Petri Net 2= " + B);
        System.out.println("Irisan = " + Irisan);
        double atas = Irisan;
        double bawah = A + B;
        Dc2 = 2 * (atas / bawah);
        System.out.println("Dice's Coefficient Similarity = " +
Dc2);
        return Dc1;
    }
```

Kode sumber 4.16 Perhitungan *Dice Coefficient Similarity*

d. Metode *Cosine Coefficient Similarity*

Perhitungan algoritma *Cosine Coefficient Similarity* secara *behavioral* adalah mencari kemiripan dengan membandingkan *intersect* objek dengan perkalian akar dari kedua objek. Algoritma *cosine* yang digunakan dalam mencari kemiripan secara behavioral sama dengan secara strukrural.

Perhitungan *cosine* ditangani oleh *method* `Cosine(int A, int B, int Irisan)` yang ditunjukkan pada **Kode sumber 4.17** di bawah ini:

```
public double Cosine(int A, int B, int Irisan) {
System.out.println("Cosine Coefficient Similarity");
        System.out.println("Petri Net 1= " + A);
        System.out.println("Petri Net 2= " + B);
        System.out.println("Irisan = " + Irisan);
        double atas = Irisan;
        double bawah = Math.sqrt(A) * Math.sqrt(B);
        Co1 = atas / bawah;
        System.out.println("Cosine Coefficient = " + Co2);
        return Co2;
    }
```

Kode sumber 4.17 Perhitungan *Cosine Coefficient Similarity*

4.4 Skalabilitas Proses Bisnis

Skalabilitas proses bisnis adalah perhitungan dengan mencari nilai *scalabiltiy* dari dua model proses bisnis yang dibandingkan. Nilai *scalability* dapat diukur dengan melibatkan beberapa parameter yaitu: *structural similarity*, *behavioral similarity*, skala model, *complexity* dan jumlah elemen dalam model proses bisnis (Ainul, et al., 2016). Sebelum melakukan perhitungan *scalability* pada model PNML, dilakukan terlebih dahulu perhitungan *complexity* atau *Control Flow Complexity* (*CFC*) pada masing-masing model PNML. Perhitungan CFC adalah menghitung jumlah logika AND-*split* dan XOR-*split* yang ada di dalam model PNML. Model PNML yang berbasis xml diambil dari direktori, kemudian di-*parsing* dengan Java DOM Parser. Elemen yang ada di dalam model PNML ditampung dalam *ArrayList*. Kemudian, membuat *class Vector* variabel *place* dan transisi yang nantinya digunakan sebagai pencari logika AND-*split* dan XOR-*split*. Selanjutnya dilakukan pemanggilan beberapa elemen menggunakan *getAtrribut* dan *TagName* untuk mengetahui isi elemen apa saja yang terdapat pada model PNML. Berikut ditunjukkan pada **Kode sumber 4.18** di bawah ini:

```
File inputFile = new File(model);
DocumentBuilderFactory dbFactory =
DocumentBuilderFactory.newInstance();
DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
Document doc = dBuilder.parse(inputFile);
doc.getDocumentElement().normalize();
System.out.println("-----------------------------");
System.out.println("Root Element :" + doc.getDocumentElement());

NodeList nList = doc.getElementsByTagName("arc");

System.out.println("-----------------------------");

ArrayList<pnml.Elemen> arrayList = new ArrayList<>();
Vector p = new Vector();
Vector t = new Vector();

for (int temp = 0; temp < nList.getLength(); temp++) {

     Node nNode = nList.item(temp);

     if (nNode.getNodeType() == Node.ELEMENT_NODE) {
          Element eElement = (Element) nNode;
          pnml.Elemen elemen = new  pnml.Elemen(eElement.
getAttribute("id"),eElement.getAttribute("source"),
eElement.getAttribute("target"),eElement.getElementsByTagName("text").i
tem(0).getTextContent());
        arrayList.add(elemen);
        if(eElement.getAttribute("source").contains("p")){
           p.add(eElement.getAttribute("source"));
        }
        if(eElement.getAttribute("source").contains("t")){
           t.add(eElement.getAttribute("source"));
        }
        if(eElement.getAttribute("target").contains("p")){
           p.add(eElement.getAttribute("target"));
        }
        if(eElement.getAttribute("target").contains("t")){
           t.add(eElement.getAttribute("target"));
                  }

               }

           }
```

Kode sumber 4.18 Pemanggilan Elemen dengan *getAttribut* dan *TagName*

Class Vector place dan transisi kemudian ditampung pada *LinkedHashset* untuk membuat matriks *place* dan transisi dari model PNML yang di*input*kan. Matriks tersebut merepresentasikan logika XOR-*split* dengan ditemukan elemen dimana "*source*" pada *place* dan "target" pada elemen transisi, sedangkan logika AND-*split* dimana "*source*" pada elemen transisi dan "target" pada elemen *place*. Berikut **Kode sumber 4.19** yang ditunjukkan di bawah ini:

```
p = new Vector(new LinkedHashSet(p));
t = new Vector(new LinkedHashSet(t));

int Model[][] = new int[p.size()][t.size()];
for (int i = 0; i < arrayList.size(); i++) {
     pnml.Elemen elemen = arrayList.get(i);
     String source = elemen.getSource();
     String target = elemen.getTarget();
     for (int j = 0; j < p.size(); j++) {
         for (int k = 0; k < t.size(); k++) {
             if (source.equals(p.get(j)) && target.equals(t.get(k))) {
                 Model[j][k] = 2;
             }
             if (source.equals(t.get(k)) && target.equals(p.get(j))) {
                 Model[j][k] = 1;
             }
         }
     }
}
System.out.print("   ");
for (int i = 0; i < t.size(); i++) {
     System.out.print(t.get(i) + " ");
}
System.out.println("");
for (int i = 0; i < Model[0].length; i++) {
     System.out.print(p.get(i) + " ");
     for (int j = 0; j < Model[1].length; j++) {
          System.out.print(Model[i][j] + "   ");
     }
     System.out.println("");
}
```

Kode sumber 4.19 Membuat Matrik dari *Input*-an Model PNML

Berdasarkan **Kode sumber 4.19** di atas, menghasilkan *output* matriks dari model PNML seperti pada **Gambar 4.6** di bawah ini:

```
Root Element :[pnml: null]
---------------------------
   t5 t9 t6 t7 t8 t2 t1 t14 t3 t15
p6 1  0  2  0  0  0  0  0  0  0
p8 0  1  0  0  0  0  0  0  0  0
p5 2  0  0  0  0  0  0  0  0  1
p9 0  2  0  0  1  0  0  0  0  0
p7 0  0  1  2  0  0  0  0  0  0
p10 0  0  0  1  2  0  0  0  0  0
p1 0  0  0  0  0  2  2  0  0  0
p2 0  0  0  0  0  1  1  2  0  0
p3 0  0  0  0  0  0  0  1  2  0
p4 0  0  0  0  0  0  0  0  1  2
---------------------------
```

Gambar 4.6 *Output* berupa Matriks dengan *Input*-an Model PNML

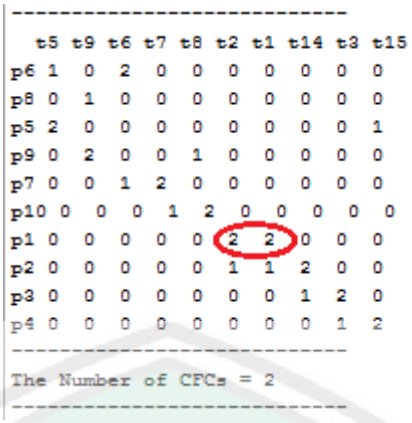
Cara menghitung logika XOR-*split* adalah dengan "*source*" pada elemen *place* dengan "target" pada elemen transisi. Jumlah XOR-*split* sesuai dengan jumlah target pada elemen transisi, sedangkan AND-*split* adalah jika terdapat "*source*"

pada elemen transisi dengan “target” pada elemen *place*. Jumlah AND-*split* akan tetap bernilai “satu” meskipun elemen transisi bercabang lebih dari satu ke elemen *place*. Jika ditemukan model PNML *sequence* artinya tidak bercabang sama sekali seperti pada **Gambar 2.3**, maka *complexity* atau CFC akan langsung bernilai “satu”. Contoh model PNML yang digunakan adalah proses bisnis Akademik *Modern* pada **Gambar 4.1**. Berikut **Kode sumber 4.20** untuk menghitung nilai CFC model PNML ditunjukkan di bawah ini:

```
int xorSplit = 0;
int andSplit = 0;
int sequence = 0;
for (int i = 0; i < Model[0].length; i++) {
      int xor = 0;
      int and = 0;
      for (int j = 0; j < Model[1].length; j++) {
           if (Model[i][j] == 2) {
               xor += 1;
           }
           if (Model[j][i] == 1) {
               and += 1;
           }
      }
      if (xor >= 2) {
          xorSplit += xor;
      }
      if (and >= 2) {
          andSplit += 1;
      }
}
if (((xorSplit == 0) & (andSplit == 0))) {
    sequence = 1;
}
System.out.println("----------------------------");
CFC1 = xorSplit + andSplit + sequence;
System.out.println("CFC = " + CFC1);
System.out.println("----------------------------");
```

Kode sumber 4.20 Perhitungan Jumlah *Control Flow Complexity*

Berdasarkan **Kode sumber 4.20** di atas, *output* berupa jumlah CFC akan diketahui. Ditunjukkan seperti pada **Gambar 4.7** di bawah ini:



```
----------------------------
  t5 t9 t6 t7 t8 t2 t1 t14 t3 t15
p6 1  0  2  0  0  0  0  0  0  0
p8 0  1  0  0  0  0  0  0  0  0
p5 2  0  0  0  0  0  0  0  0  1
p9 0  2  0  0  1  0  0  0  0  0
p7 0  0  1  2  0  0  0  0  0  0
p10 0  0  0  1  2  0  0  0  0  0
p1 0  0  0  0  0  2  2  0  0  0
p2 0  0  0  0  0  1  1  2  0  0
p3 0  0  0  0  0  0  0  1  2  0
p4 0  0  0  0  0  0  0  0  1  2
----------------------------
The Number of CFCs = 2
----------------------------
```

Gambar 4.7 *Output* Perhitungan CFC

Berdasarkan dari **Gambar 4.7** di atas, bahwa CFC bernilai "dua" karena elemen *place* bercabang ke dua buah elemen transisi.

Selanjutnya, menghitung nilai *scalability* berdasarkan parameter yang dibutuhkan oleh algoritma *scalability* yang terdapat pada persamaan 2.7 seperti di bawah ini:

$$scalability(A,B) = 1 - \big(skala(A,B) * sim(A,B)\big)$$

Perhitungan *scalability* ditangani oleh *method* `skalability(int A, int B, int CFC11, int CFC22)`. Berikut kode sumber yang digunakan untuk menghitung nilai *scalability* pada dua model PNML yang ditunjukkan pada **Kode sumber 4.21** di bawah ini:

```
public double skalability(int A, int B, int CFC11, int CFC22) {

    System.out.println("Skalabilitas");
    double SkalaA = jumlahNode1 * CFC1;
    double SkalaB = jumlahNode2 * CFC2;
    double SkalaFix = SkalaA / SkalaB;
    double scalability = 1 - (SkalaFix * rata2);
    System.out.println("Scalability of Two Model= " + scalability);

     return scalability;

}
```

Kode sumber 4.21 Perhitungan *Scalability*

4.5 Simulasi Pertumbuhan Proses Bisnis

Simulasi pertumbuhan proses bisnis pada aplikasi ini menggunakan *Production Rule Cellular Automata*. Aturan produksi mengikuti seperti di bawah ini:

1. Aturan produksi $\alpha \rightarrow \beta$ yang diterapkan pada suatu string $w = \alpha\alpha c$ mengganti kemunculan α menjadi β sehingga string tersebut menjadi $w = \alpha\beta c$ sehingga daat ditulis $\alpha\alpha c \Rightarrow \alpha\beta c$ ($\alpha\alpha c$ *memproduksi* $\alpha\beta c$).
2. Produksi tersebut dapat diterapkan berkali-kali

$$w \Rightarrow w_2 \Rightarrow w_3 \Rightarrow w_n$$

3. Atau dapat ditulis $w_1 \Rightarrow^* w_n$, jika minimal harus ada 1 aturan produksi yang ditetapkan : $w_1 \Rightarrow^+ w_n$

Aturan di atas diterapkan dalam pembuatan *proposed algorithm* seperti pada **Gambar 3.22**.

Langkah untuk simulasi pertumbuhan proses bisnis pada penelitian ini sebagai berikut: **Pertama**, meng-*input*-kan dua data uji dari tipe pondok yang berbeda dengan bagian yang sama. Misal: **Proses Bisnis Akademik *Modern*** sebagai *input*-an A dibandingkan dengan **Proses Bisnis Akademik Mahasiswa** sebagai *input*-an B. **Kedua**, menghitung *structural* dan *behavioral similarity* dari kedua model PNML tersebut. **Ketiga**, menghitung nilai *scalability* dari kedua *input*-an model PNML. Jika nilai *scalability* yang ditemukan pada perhitungan pertama adalah " >= 0" dan "< 1" maka dilakukan simulasi pertumbuhan. Tetapi penulis membuat syarat tambahan yaitu jika *scalability* pada iterasi ke-0 atau *scalability* awal bernilai "<0.1", maka simulasi pertumbuhan tidak terjadi. Terdapat kriteria dalam

menentukan *input*-an A dan *input*-an B. *Input*-an A harus mempunyai nilai *complexity* yang lebih rendah dibandingkan dengan *input*-an B.

Pertumbuhan proses bisnis yang dimaksud adalah dengan menumbuhkan tiga elemen / *node* pada model PNML *input*-an A. Tiga *node* tersebut yaitu: *arc*, *place* dan transisi. Berikut di bawah ini **Kode sumber 4.22** adalah untuk membuat elemen / *node place* yang ditangani oleh *method* `place(String id, Document doc, Element net, String offsetX, String offSetY, String positionX, String positionY)` sebagai berikut:

```
public Node place(String id, Document doc, Element net, String offsetX,
String offSetY, String positionX, String positionY) {
    // place element
   Element place = doc.createElement("place");
   Attr attrType = doc.createAttribute("id");
   attrType.setValue(id);
   place.setAttributeNode(attrType);
   net.appendChild(place);

   Element name = doc.createElement("name");
   place.appendChild(name);

   Element grafik = doc.createElement("graphics");
   place.appendChild(grafik);

   Element text = doc.createElement("text");
   text.appendChild(doc.createTextNode(id));
   name.appendChild(text);

   Element grafikName = doc.createElement("graphics");
   name.appendChild(grafikName);

   Element offset = doc.createElement("offset");
   Attr attrTypeOffSet = doc.createAttribute("x");
   attrTypeOffSet.setValue(offsetX);
   offset.setAttributeNode(attrTypeOffSet);
   attrTypeOffSet = doc.createAttribute("y");
   attrTypeOffSet.setValue(offSetY);
   offset.setAttributeNode(attrTypeOffSet);
   grafikName.appendChild(offset);

   Element position = doc.createElement("position");
   Attr attrTypePosition = doc.createAttribute("x");
   attrTypePosition.setValue(positionX);
   position.setAttributeNode(attrTypePosition);
   attrTypePosition = doc.createAttribute("y");
   attrTypePosition.setValue(positionY);
   position.setAttributeNode(attrTypePosition);
   grafik.appendChild(position);

   Element dimention = doc.createElement("dimention");
   Attr attrTypeDimention = doc.createAttribute("x");
   attrTypeDimention.setValue("40");
```

```
    dimention.setAttributeNode(attrTypeDimention);
    attrTypeDimention = doc.createAttribute("y");
    attrTypeDimention.setValue("40");
    dimention.setAttributeNode(attrTypeDimention);
    grafik.appendChild(dimention);
    return place;
}
```

Kode sumber 4.22 Membuat Elemen *Place*

Berdasarkan **Kode sumber 4.22** di atas, membuat elemen *place* dengan *method* objek `Node`.

Selanjutnya, membuat elemen transisi yang ditangani oleh *method* `transisi(String id, Document doc, Element net, String offsetX, String offSetY, String positionX, String positionY)`. Berikut **Kode sumber 4.23** untuk membuat elemen transisi seperti di bawah ini:

```
public Node transisi(String id, Document doc, Element net, String
offsetX, String offSetY, String positionX, String positionY) {
    // place element
     Element transisi = doc.createElement("transition")
     Attr attrType = doc.createAttribute("id");
     attrType.setValue(id);
     transisi.setAttributeNode(attrType);
     net.appendChild(transisi);

     Element name = doc.createElement("name");
     transisi.appendChild(name);

     Element grafik = doc.createElement("graphics");
     transisi.appendChild(grafik);

     Element text = doc.createElement("text");
     text.appendChild(
                 doc.createTextNode(id));
     name.appendChild(text);

     Element grafikName = doc.createElement("graphics");
     name.appendChild(grafikName);

     Element offset = doc.createElement("offset");
     Attr attrTypeOffSet = doc.createAttribute("x");
     attrTypeOffSet.setValue(offsetX);
     offset.setAttributeNode(attrTypeOffSet);
     attrTypeOffSet = doc.createAttribute("y");
     attrTypeOffSet.setValue(offSetY);
     offset.setAttributeNode(attrTypeOffSet);
     grafikName.appendChild(offset);

     Element position = doc.createElement("position");
     Attr attrTypePosition = doc.createAttribute("x");
     attrTypePosition.setValue(positionX);
     position.setAttributeNode(attrTypePosition);
     attrTypePosition = doc.createAttribute("y");
     attrTypePosition.setValue(positionY);
```

```
    position.setAttributeNode(attrTypePosition);
    grafik.appendChild(position);

    Element dimention = doc.createElement("dimention");
    Attr attrTypeDimention = doc.createAttribute("x");
    attrTypeDimention.setValue("40");
    dimention.setAttributeNode(attrTypeDimention);
    attrTypeDimention = doc.createAttribute("y");
    attrTypeDimention.setValue("40");
    dimention.setAttributeNode(attrTypeDimention);
    grafik.appendChild(dimention);

    Element toolspecific = doc.createElement("toolspecific");
    Attr attrTypeToolSpe = doc.createAttribute("tool");
    attrTypeToolSpe.setValue("WoPed");
    toolspecific.setAttributeNode(attrTypeToolSpe);
    attrTypeToolSpe = doc.createAttribute("version");
    attrTypeToolSpe.setValue("1.0");
    toolspecific.setAttributeNode(attrTypeToolSpe);
    transisi.appendChild(toolspecific);

    Element time = doc.createElement("time");
    time.appendChild(doc.createTextNode("0"));
    toolspecific.appendChild(time);

    Element timeUnit = doc.createElement("timeUnit");
    timeUnit.appendChild(doc.createTextNode("1"));
    toolspecific.appendChild(timeUnit);

    Element orientation = doc.createElement("orientation");
    orientation.appendChild(doc.createTextNode("1"));
    toolspecific.appendChild(orientation);
    return transisi;

}
```

Kode sumber 4.23 Membuat Elemen Transisi

Elemen *arc* dibuat ditangani oleh *method* `arc(String id, Document doc, Element net, String source, String target)`. Berikut **Kode sumber 4.24** yang berkaitan dengan pembuatan elemen *arc* seperti di bawah ini:

```
public Node arc(String id, Document doc, Element net, String source,
String target) {
    // place element
   Element arc = doc.createElement("arc");
   Attr attrType = doc.createAttribute("id");
   attrType.setValue(id);
   arc.setAttributeNode(attrType);
   attrType = doc.createAttribute("source");
   attrType.setValue(source);
   arc.setAttributeNode(attrType);
   attrType = doc.createAttribute("target");
   attrType.setValue(target);
   arc.setAttributeNode(attrType);
   net.appendChild(arc);


   Element inscription = doc.createElement("inscription");
```

```
    arc.appendChild(inscription);

    Element grafik = doc.createElement("graphics");
    arc.appendChild(grafik);

     Element text = doc.createElement("text");
     text.appendChild(doc.createTextNode("1"));
     inscription.appendChild(text);

     Element toolspecific = doc.createElement("toolspecific");
     Attr attrTypeToolSpe = doc.createAttribute("tool");
     attrTypeToolSpe.setValue("WoPed");
     toolspecific.setAttributeNode(attrTypeToolSpe);
     attrTypeToolSpe = doc.createAttribute("version");
     attrTypeToolSpe.setValue("1.0");
     toolspecific.setAttributeNode(attrTypeToolSpe);
     arc.appendChild(toolspecific);

     Element probability = doc.createElement("probability");
     probability.appendChild(doc.createTextNode("1.0"));
     toolspecific.appendChild(probability);

     Element displayProbabilityOn =
doc.createElement("displayProbabilityOn");
 displayProbabilityOn.appendChild(doc.createTextNode("false"));
     toolspecific.appendChild(displayProbabilityOn);

     Element displayProbabilityPosition =
doc.createElement("displayProbabilityPosition");
     Attr attrTypedisplayProbabilityPosition = doc.createAttribute("x");
     attrTypedisplayProbabilityPosition.setValue("500.0");

displayProbabilityPosition.setAttributeNode(attrTypedisplayProbabilityP
osition);
     attrTypedisplayProbabilityPosition =  doc.createAttribute("y");

     attrTypedisplayProbabilityPosition.setValue("0.0");

displayProbabilityPosition.setAttributeNode(attrTypedisplayProbabilityP
osition); toolspecific.appendChild(displayProbabilityPosition);
     return arc;

}
```

Kode sumber 4.24 Membuat Elemen *Arc*

Pertumbuhan yang dimaksud adalah menumbuhkan *node* baru pada proses bisnis yang akan ditumbuhkan jika nilai *scalability* ">= 0" dan "< 1". Terdapat tiga *node* atau elemen yang akan ditumbuhkan pada model *Petri net* yaitu *place*, *arc* dan transisi.

Pola pertumbuhan proses bisnis dapat dilakukan dengan dua pembobotan pada proses simulasi pertumbuhan proses bisnis pada penelitian ini yaitu:

1. Pembobotan percabangan

Pembobotan percabangan yang dimaksudkan adalah simulasi pertumbuhan elemen pada model *Petri net* cenderung ke arah percabangan. Berikut kode sumber untuk *create node* yang lebih ke pembobotan percabangan yang ditunjukkan pada **Kode sumber 4.25** di bawah ini:

```
ParserStructural ps = new ParserStructural();
ps.Parsing1(pathmodel1);
ps.ControlFlow1(pathmodel1);
ps.ControlFlow2(pathmodel2);
if ((scalability > 0)&& (scalability > 0.3) && (scalability < 1)) {

     ps.XML(pathmodel1);
     System.out.println("Right");

    } else if ((scalability < 0.3) && (scalability > 0) &&
(scalability < 1)) {

      ps.XML1(pathmodel1);
      System.out.println("Benar Juga");

     }
     else {
     JOptionPane.showMessageDialog(null, "The Growth of Business
Process Simulation Stopped");
}
```

Kode sumber 4.25 Logika Pembobotan Percabangan

Berdasarkan **Kode sumber 4.25** di atas, terdapat *scalability* ">0.3" akan di eksekusi oleh *class* `XML(pathmodel1)`. Sedangkan jika *scalability* "<0.3" di eksekusi *class* `XML1(pathmodel1)`. *Class* `XML(pathmodel1)` digunakan untuk membuat elemen *place* yang nantinya akan ditumbuhkan pada model *Petri net*. Berikut **Kode sumber 4.26** untuk *create* elemen *place* seperti di bawah ini:

```
NodeList p = doc.getElementsByTagName("place");
NodeList t = doc.getElementsByTagName("transition");
NodeList a = doc.getElementsByTagName("arc");;


//Elemen Lama Random Split
String[] p_lamas = {"p1", "p2", "p3", "p4"};
String[] t_lamas = {"t1", "t2", "t3", "t4"};
   String random_plamas = (p_lamas[new
Random().nextInt(p_lamas.length)]);
   String random_tlamas = (t_lamas[new
Random().nextInt(t_lamas.length)]);
```

```
//Elemen Lama Random Join
String[] p_lamaj = {"p3", "p4", "p5", "p6", "p7"};
String[] t_lamaj = {"t3", "t4", "t5", "t6", "t7"};
   String random_plamaj = (p_lamaj[new
Random().nextInt(p_lamaj.length)]);
   String random_tlamaj = (t_lamaj[new
Random().nextInt(t_lamaj.length)]);

//Element Baru Random
   String[] place = {"pa", "pb", "pc", "pd", "pe", "pf", "pg", "ph",
"pi"};
   String[] transisi = {"ta", "tb", "tc", "td", "te", "tf", "tg",
"th", "ti"};
   String random_pbaru = (place[new Random().nextInt(place.length)]);
   String random_tbaru = (transisi[new
Random().nextInt(transisi.length)]);

if (t.equals(t)) {
    p.item(0).getParentNode().insertBefore(place(random_pbaru, doc,
net, "140", "150", "140", "110"), p.item(0));
    a.item(0).getParentNode().insertBefore(arc("a1", doc, net,
random_tlamas, random_pbaru), a.item(0));
    a.item(0).getParentNode().insertBefore(arc("a1", doc, net,
random_pbaru, random_tlamaj), a.item(0));

   }
```

Kode sumber 4.26 *Method* XML (pathmodel1)

Menurut **Kode sumber 4.26** di atas, dalam *class* `XML(pathmodel1)` akan langsung men-*create* tiga elemen yaitu: *place*, *arc* (untuk *split*) dan *arc* (untuk *join*). Penamaan elemen *place* baru telah disediakan di dalam varibel *String* yang bersifat *random* artinya nama dari elemen *place* akan mengambil salah satu nama di dalam *Array*. Maka, penempatan ketiga elemen tersebut ditumbuhkan pada model *Petri net* secara *random*. *Arc-split* untuk menghubungkan transisi lama dari model *Petri net* dengan *place* baru, sedangkan *arc-join* digunakan untuk menghubungkan *place* baru dengan transisi lama yang lain.

`Class XML1(pathmodel1)` digunakan untuk membuat elemen transisi yang nantinya akan ditumbuhkan pada model *Petri net*. Berikut **Kode sumber 4.27** untuk *create* elemen transisi seperti di bawah ini:

```
NodeList p = doc.getElementsByTagName("place");
NodeList t = doc.getElementsByTagName("transition");
```

```
NodeList a = doc.getElementsByTagName("arc");;


//Elemen Lama Random Split
String[] p_lamas = {"p1", "p2", "p3", "p4"};
String[] t_lamas = {"t1", "t2", "t3", "t4"};
   String random_plamas = (p_lamas[new
Random().nextInt(p_lamas.length)]);
   String random_tlamas = (t_lamas[new
Random().nextInt(t_lamas.length)]);

//Elemen Lama Random Join
String[] p_lamaj = {"p3", "p4", "p5", "p6", "p7"};
String[] t_lamaj = {"t3", "t4", "t5", "t6", "t7"};
   String random_plamaj = (p_lamaj[new
Random().nextInt(p_lamaj.length)]);
   String random_tlamaj = (t_lamaj[new
Random().nextInt(t_lamaj.length)]);

//Element Baru Random
   String[] place = {"pa", "pb", "pc", "pd", "pe", "pf", "pg", "ph",
"pi"};
   String[] transisi = {"ta", "tb", "tc", "td", "te", "tf", "tg",
"th", "ti"};
   String random_pbaru = (place[new Random().nextInt(place.length)]);
   String random_tbaru = (transisi[new
Random().nextInt(transisi.length)]);

if ((p.equals(p))) {
   t.item(0).getParentNode().insertBefore(transisi(random_tbaru, doc,
net, "140", "150", "140", "110"), t.item(0));
    a.item(0).getParentNode().insertBefore(arc("a1", doc, net,
random_plamas, random_tbaru), a.item(0));
    a.item(0).getParentNode().insertBefore(arc("a1", doc, net,
random_tbaru, random_plamaj), a.item(0));

   }
```

Kode sumber 4.27 *Method* XML1 (pathmodel1)

Menurut **Kode sumber 4.27** di atas, dalam *class* `XML1(pathmodel1)` akan langsung men-*create* tiga elemen yaitu: transisi, *arc* (untuk *split*) dan *arc* (untuk *join*). Penamaan elemen transisi baru telah disediakan di dalam varibel *String* yang bersifat *random* artinya nama dari elemen transisi akan mengambil salah satu nama di dalam *Array*. Maka, penempatan ketiga elemen tersebut ditumbuhkan pada model PNML secara *random*. *Arc-split* untuk menghubungkan *place* lama dari model PNML dengan transisi baru, sedangkan *arc-join* digunakan untuk menghubungkan transisi baru dengan *place* lama yang lain.

2. Pembobotan *Sequence*

Pembobotan sequence yang dimaksudkan adalah simulasi pertumbuhan elemen pada model PNML cenderung ke arah *sequence* / tanpa adanya percabangan dalam penumbuhan elemen baru. Berikut kode sumber untuk *create node* yang lebih ke pembobotan *sequence* yang ditunjukkan pada **Kode sumber 4.28** di bawah ini:

```
ParserStructural ps = new ParserStructural();
ps.Parsing1(pathmodel1);
ps.ControlFlow1(pathmodel1);
ps.ControlFlow2(pathmodel2);
if ((scalability > 0) && (scalability < 1)) {

    ps.XMLSeq(pathmodel1);
    System.out.println("Right");

   }
    else {
    JOptionPane.showMessageDialog(null, "The Growth of Business
Process Simulation Stopped");
}
```

Kode sumber 4.28 Logika Pembobotan *Sequence*

Berdasarkan **Kode sumber 4.28** di atas, terdapat *scalability* ">0" akan di eksekusi oleh *class* `XMLSeq(pathmodel1)`. *Class* `XMLSeq(pathmodel1)` digunakan untuk membuat elemen *place* dan transisi yang nantinya akan ditumbuhkan pada model PNML. Berikut **Kode sumber 4.29** untuk *create* elemen transisi dan *place* seperti di bawah ini:

```
NodeList p = doc.getElementsByTagName("place");
NodeList t = doc.getElementsByTagName("transition");
NodeList a = doc.getElementsByTagName("arc");;

//Elemen Lama Random Split
String[] p_lamas = {"p1", "p2", "p3", "p4"};
String[] t_lamas = {"t1", "t2", "t3", "t4"};
   String random_plamas = (p_lamas[new
Random().nextInt(p_lamas.length)]);
   String random_tlamas = (t_lamas[new
Random().nextInt(t_lamas.length)]);

//Elemen Lama Random Join
String[] p_lamaj = {"p3", "p4", "p5", "p6", "p7"};
String[] t_lamaj = {"t3", "t4", "t5", "t6", "t7"};
```

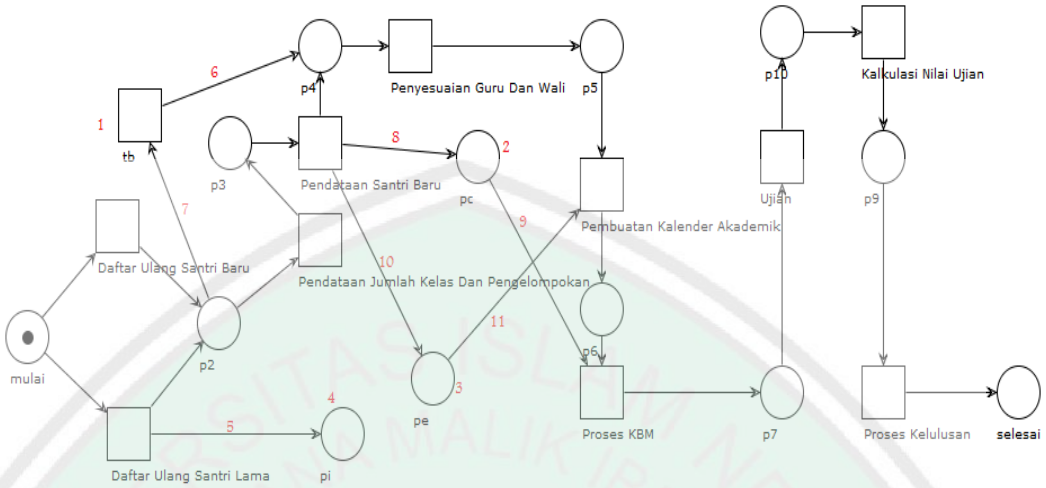
```
   String random_plamaj = (p_lamaj[new
Random().nextInt(p_lamaj.length)]);
   String random_tlamaj = (t_lamaj[new
Random().nextInt(t_lamaj.length)]);

//Element Baru Random
   String[] place = {"pa", "pb", "pc", "pd", "pe", "pf", "pg", "ph",
"pi"};
   String[] transisi = {"ta", "tb", "tc", "td", "te", "tf", "tg",
"th", "ti"};
   String random_pbaru = (place[new Random().nextInt(place.length)]);
   String random_tbaru = (transisi[new
Random().nextInt(transisi.length)]);

if (t.equals(t)) {

    p.item(0).getParentNode().insertBefore(place(random_pbaru, doc,
net, "140", "150", "140", "110"), p.item(0));
    a.item(0).getParentNode().insertBefore(arc("a1", doc, net,
random_tlamas, random_pbaru), a.item(0));
    a.item(0).getParentNode().insertBefore(arc("a1", doc, net,
random_pbaru, random_tbaru), a.item(0));

}

if (p.equals(p)){
   t.item(0).getParentNode().insertBefore(transisi(random_tbaru, doc,
net, "140", "150", "140", "110"), t.item(0));
   a.item(0).getParentNode().insertBefore(arc("a1", doc, net,
random_tbaru, random_plamaj), a.item(0));

}
```

Kode sumber 4.29 *Method* XMLSeq (pathmodel1)

Menurut **Kode sumber 4.29** di atas, dalam *class* `XMLSeq(pathmodel1)` akan langsung men-*create* lima elemen yaitu: transisi, *place*, *arc1*, *arc2* dan *arc3*. Penamaan elemen transisi dan *place* baru telah disediakan di dalam varibel *String* yang bersifat *random* artinya nama dari elemen transisi dan *place* akan mengambil salah satu nama di dalam *Array*. Maka, penempatan kelima elemen tersebut ditumbuhkan pada model PNML secara *random*. *Arc1* untuk menghubungkan transisi lama dari model PNML dengan *place* baru, *arc2* digunakan untuk menghubungkan *place* baru dengan transisi baru. Sedangkan *arc3* digunakan untuk menghubungkan transisi baru dengan *place* lama.

Output dari simulasi pertumbuhan model proses bisnis adalah file *Petri net*. Berikut hasil file *Petri net* hasil dari pertumbuhan proses bisnis pada **Gambar 4.1**

dengan menggunakan teori *Production Rule Cellular Automata* seperti pada **Gambar 4.8** di bawah ini:



Gambar 4.8 Output dari Simulasi Pertumbuhan Proses Bisnis

Berdasarkan **Gambar 4.8** di atas, terdapat elemen baru setelah dilakukannya simulasi pertumbuhan proses bisnis. Elemen baru pada pemodelan *Petri net* ditandai dengan penomoran bewarna merah. Penamaan pada elemen baru juga berhasil dilakukan seperti yang diharapkan. Adanya tambahan elemen baru tersebut menandai bahwa simulasi pertumbuhan berhasil dilakukan seperti yang diharapkan dalam penelitian ini.

4.6 Uji Coba Sistem

Pengujian sistem akan dilakukan pada model PNML untuk menghitung *scalability* awal, mana saja proses yang bisa ditumbuhkan dengan syarat *scalability* bernilai "> 0" dan "< 1". Berikut data uji coba sistem yang akan diuji cobakan berdasarkan bagian masing-masing diantaranya:

1. Akademik

   Dilakukan perhitungan perbandingan untuk menemukan nilai *scalability* pada bagian Akademik. Berikut ditemukan nilai *scalability* dari perbandingan

model PNML dari bagian Akademik yang ditunjukkan pada **Tabel 4.1** di bawah ini:

Tabel 4.1 Perbandingan Nilai *Scalability* Akademik

Model A \ Model B	**Akademik *Tahfidz***	**Akademik *Modern***	**Akademik Salaf**	**Akademik Mahasiswa**
Akademik *Tahfidz*	0.0	0.486051055	0.581093858	0.600400776
Akademik *Modern*	-	0.0	0.565881147	0.563700293
Akademik *Salaf*	-	-	0.0	0.208961765
Akademik Mahasiswa	-	-	-	0.0

Dari **Tabel 4.1** di atas, didapatkan variasi nilai *scalability* dengan membandingkan dua model PNML dari bagian Akademik. Jika *complexity input*-an A > *input*-an B, maka tidak dapat dilakukan simulasi pertumbuhan. Begitu juga sebaliknya, jika *complexity input*-an A < *input*-an B, maka dapat dilakukan simulasi pertumbuhan. Namun, jika pada tabel di atas terdapat kolom yang isinya tanda "-" artinya perbandingan antar dua model PNML menghasilkan nilai *scalability* tetapi tidak dapat dilakukan simulasi pertumbuhan. Jika membandingkan *input*-an A dan *input*-an B dengan model PNML yang sama, maka nilai *scalability* akan otomatis bernilai nol.

Simulasi pertumbuhan dilakukan dengan pembobotan pada percabangan dan *sequence*. Hasil simulasi pertumbuhan pada pembobotan percabangan dan *sequence* akan mendapatkan iterasi berbeda. Hal ini dikarenakan simulasi pertumbuhan elemen pada proses bisnis dilakukan secara *random*. Oleh karena itu, hasil simulasi pada percobaan pertama, kedua dan seterusnya akan berbeda. Namun, tidak menutup kemungkinan mendapatkan nilai iterasi yang sama.

Nilai *scalability* terus mengalami penurunan seiring dengan bertambahnya iterasi pertumbuhan pada proses bisnis. Iterasi pertumbuhan menyebabkan

scalability bernilai ">=0", artinya nilai *scalability* tidak sampai bernilai minus. Namun, pada iterasi terakhir, *scalability* dapat bernilai minus karena pada iterasi sebelumnya *scalability* terindentifikasi bernilai ">=0" dan akan berlanjut pertumbuhannya. Hal ini menyebabkan *scalability* pada iterasi terakhir bernilai minus. Sehingga, nilai *scalability* minimum terletak pada iterasi sebelum iterasi terakhir.

Berikut ditunjukkan hasil simulasi pertumbuhan dari Akademik *Tahfidz* sebagai *input*-an A dan Akademik *Modern* sebagai *input*-an B. Perbandingan antar model tersebut, *input*-an A mewakili proses bisnis yang bercabang, sama halnya dengan *input*-an B. Hasil perbandingan ditunjukkan pada **Tabel 4.2** di bawah ini:

Tabel 4.2 Simulasi Pertumbuhan Akademik *Tahfidz* & Akademik *Modern*

PERCABANGAN						
Percobaan 1	Model B / Model A	Akademik *Modern*	Percobaan 2	Model B / Model A	Akademik *Modern*
	Akademik *Tahfidz*	0.486051055		Akademik *Tahfidz*	0.486051055
	Iterasi 1	0.208399391		Iterasi 1	0.472266261
	Iterasi 2	-0.369560058		Iterasi 2	0.178263965
	Iterasi 3			Iterasi 3	-0.426823023
A N D	Iterasi 1	0.208399391	X O R		-
	Iterasi 2	0.178263965			-
	Iterasi 3	-0.141458418			-
SEQUENCE						
Percobaan 1	Model B / Model A	Akademik *Modern*	Percobaan 2	Model B / Model A	Akademik *Modern*
	Akademik *Tahfidz*	0.486051055		Akademik *Tahfidz*	0.486051055
	Iterasi 1	0.189969094		Iterasi 1	0.189969094
	Iterasi 2			Iterasi 2	-0.088661589

Menurut **Tabel 4.2** di atas, simulasi pertumbuhan percobaan pertama dan kedua pada pembobotan percabangan mendapatkan hasil iterasi yang berbeda. Sedangkan pada pembobotan *sequence*, percobaan pertama dan kedua mendapatkan hasil iterasi dan nilai minimum *scalability* yang sama. Dilakukan

juga simulasi pertumbuhan dengan pola AND, artinya percabangan yang lebih ke arah logika AND dan mendapatkan hasil iterasi yang berbeda. Seperti yang telah dipaparkan, nilai *scalability* awal pada perbandingan proses bisnis Akademik *Tahfidz* pada Gambar 3.17 dan proses bisnis Akademik *Modern* pada Gambar 3.7 adalah 0.486051055, maka simulasi pertumbuhan dapat terjadi.

Berikut ditunjukkan hasil simulasi pertumbuhan dari Akademik *Tahfidz* sebagai *input*-an A dan Akademik *Salaf* sebagai *input*-an B. Perbandingan antar model tersebut, *input*-an A mewakili proses bisnis yang bercabang, sama halnya dengan *input*-an B. Hasil perbandingan ditunjukkan pada **Tabel 4.3** di bawah ini:

Tabel 4.3 Simulasi Pertumbuhan Akademik *Tahfidz* & Akademik *Salaf*

PERCABANGAN					
Percobaan 1	Model B / Model A	Akademik *Salaf*	Percobaan 2	Model B / Model A	Akademik *Salaf*
	Akademik *Tahfidz*	0.581093858		Akademik *Tahfidz*	0.581093858
	Iterasi 1	0.569559434		Iterasi 1	0.354339152
	Iterasi 2	0.329225427		Iterasi 2	0.329225427
	Iterasi 3	0.067503302		Iterasi 3	0.067503302
	Iterasi 4	-0.211845644		Iterasi 4	-0.454214772
AND	Iterasi 1	0.424532928	XOR		-
	Iterasi 2	0.101256899			-
	Iterasi 3	-0.078491720			-
SEQUENCE					
Percobaan 1	Model B / Model A	Akademik *Salaf*	Percobaan 2	Model B / Model A	Akademik *Salaf*
	Akademik *Tahfidz*	0.581093858		Akademik *Tahfidz*	0.581093858
	Iterasi 1	0.338967644		Iterasi 1	0.338967644
	Iterasi 2	0.203887209		Iterasi 2	0.118871764
	Iterasi 3	0.118871764		Iterasi 3	-0.092642150
	Iterasi 4	-0.150934143		Iterasi 4	

Menurut **Tabel 4.3** di atas, simulasi pertumbuhan percobaan pertama dan kedua pada pembobotan percabangan dan *sequence* mendapatkan hasil yang

berbeda. Dilakukan juga simulasi pertumbuhan dengan pola AND, artinya percabangan yang lebih ke arah logika AND dan mendapatkan hasil iterasi yang berbeda. Seperti yang telah dipaparkan, nilai *scalability* awal pada perbandingan proses bisnis Akademik *Tahfidz* pada Gambar 3.17 dan proses bisnis Akademik *Salaf* pada Gambar 3.2 adalah 0.58109385, maka simulasi pertumbuhan dapat terjadi. Tetapi nilai *scalability* minimum pada pembotan percabangan maupun *sequence* hasilnya sama.

Berikut ditunjukkan hasil simulasi pertumbuhan dari Akademik *Tahfidz* sebagai *input*-an A dan Akademik Mahasiswa sebagai *input*-an B pada **Tabel 4.4** di bawah ini:

Tabel 4.4 Simulasi Pertumbuhan Akademik *Tahfidz* & Akademik Mahasiswa

PERCABANGAN					
Percobaan 1	Model B / Model A	Akademik Mahasiswa	Percobaan 2	Model B / Model A	Akademik Mahasiswa
	Akademik *Tahfidz*	0.600400776		**Akademik *Tahfidz***	0.600400776
	Iterasi 1	0.383847958		Iterasi 1	0.383847958
	Iterasi 2	0.157513295		Iterasi 2	0.368134971
	Iterasi 3	-0.309533179		Iterasi 3	0.126977880
	Iterasi 4			Iterasi 4	-0.134831846
A N D	Iterasi 1	0.589231972	X O R		-
	Iterasi 2	0.368134971			-
	Iterasi 3	-0.309533179			-
SEQUENCE					
Percobaan 1	Model B / Model A	Akademik Mahasiswa	Percobaan 2	Model B / Model A	Akademik Mahasiswa
	Akademik *Tahfidz*	0.600400776		**Akademik *Tahfidz***	0.600400776
	Iterasi 1	0.373264633		Iterasi 1	0.373264633
	Iterasi 2	0.126977880		Iterasi 2	0.126977880
	Iterasi 3	0.125767119		Iterasi 3	-0.306849320
	Iterasi 4	-0.408070497		Iterasi 4	

Menurut **Tabel 4.4** di atas, simulasi pertumbuhan percobaan pertama dan kedua pada pembobotan percabangan dan *sequence* mendapatkan hasil yang berbeda. Dilakukan juga simulasi pertumbuhan dengan pola AND, artinya

percabangan yang lebih ke arah logika AND dan mendapatkan hasil iterasi yang berbeda. Seperti yang telah dipaparkan, nilai *scalability* awal pada perbandingan proses bisnis Akademik *Tahfidz* pada Gambar 3.17 dan proses bisnis Akademik Mahasiswa pada Gambar 3.12 adalah 0.600400776, maka simulasi pertumbuhan dapat terjadi.

Berikut ditunjukkan hasil simulasi pertumbuhan dari Akademik *Modern* sebagai *input*-an A dan Akademik *Salaf* sebagai *input*-an B pada **Tabel 4.5** di bawah ini: X

Tabel 4.5 Simulasi Pertumbuhan Akademik *Modern* & Akademik *Salaf*

PERCABANGAN					
Percobaan 1	Model B / Model A	Akademik *Salaf*	Percobaan 2	Model B / Model A	Akademik *Salaf*
	Akademik *Modern*	0.565881147		Akademik *Modern*	0.565881147
	Iterasi 1	0.319590706		Iterasi 1	0.546393804
	Iterasi 2	0.291291366		Iterasi 2	0.291291366
	Iterasi 3	0.083784589		Iterasi 3	0.083784581
	Iterasi 4	-0.145269263		Iterasi 4	-0.106082848
AND	Iterasi 1	0.317590706	XOR		-
	Iterasi 2	0.282291366			-
	Iterasi 3	-0.145269263			-
SEQUENCE					
Percobaan 1	Model B / Model A	Akademik *Salaf*	Percobaan 2	Model B / Model A	Akademik *Salaf*
	Akademik *Modern*	0.565881147		Akademik *Modern*	0.565881147
	Iterasi 1	0.348102792		Iterasi 1	0.348102792
	Iterasi 2	-0.0790778007		Iterasi 2	0.136737759
	Iterasi 3			Iterasi 3	0.091510681

Menurut **Tabel 4.5** di atas, simulasi pertumbuhan percobaan pertama dan kedua pada pembobotan percabangan dan *sequence* mendapatkan hasil yang berbeda. Dilakukan juga simulasi pertumbuhan dengan pola AND, artinya percabangan yang lebih ke arah logika AND dan mendapatkan hasil iterasi yang berbeda. Seperti yang telah dipaparkan, nilai *scalability* awal pada perbandingan proses bisnis Akademik *Modern* pada Gambar 3.7 dan proses

bisnis Akademik *Salaf* pada Gambar 3.2 adalah 0.5658811475409837, maka simulasi pertumbuhan dapat terjadi.

Berikut ditunjukkan hasil simulasi pertumbuhan dari Akademik *Modern* sebagai *input*-an A dan Akademik Mahasiswa sebagai *input*-an B pada **Tabel 4.6** di bawah ini:

Tabel 4.6 Simulasi Pertumbuhan Akademik Modern & Akademik Mahasiswa

PERCABANGAN					
Percobaan 1	Model B / Model A	Akademik Mahasiswa	Percobaan 2	Model B / Model A	Akademik Mahasiswa
	Akademik *Modern*	0.563700293		**Akademik *Modern***	0.563700293
	Iterasi 1	0.329427549		Iterasi 1	0.552951699
	Iterasi 2	0.301461581		Iterasi 2	0.301461581
	Iterasi 3	0.319100892		Iterasi 3	0.319100892
	Iterasi 4	0.118032610		Iterasi 4	0.118032610
	Iterasi 5	-0.354721824		Iterasi 5	0.096852116
	Iterasi 6			Iterasi 6	-0.528625908
AND	Iterasi 1	0.552951699	XOR		-
	Iterasi 2	0.301461581			-
	Iterasi 3	0.263377325			-
	Iterasi 4	0.116052791			-
	Iterasi 5	-0.031271743			-
SEQUENCE					
Percobaan 1	Model B / Model A	Akademik Mahasiswa	Percobaan 2	Model B / Model A	Akademik Mahasiswa
	Akademik *Modern*	0.563700293		**Akademik *Modern***	0.563700293
	Iterasi 1	0.320006970		Iterasi 1	0.320006970
	Iterasi 2	0.156247247		Iterasi 2	0.176716277
	Iterasi 3	0.092134523		Iterasi 3	0.156247247
	Iterasi 4	-0.234925583		Iterasi 4	0.092134523

Menurut **Tabel 4.6** di atas, simulasi pertumbuhan percobaan pertama dan kedua pada pembobotan percabangan dan *sequence* mendapatkan hasil yang berbeda. Dilakukan juga simulasi pertumbuhan dengan pola AND, artinya percabangan yang lebih ke arah logika AND dan mendapatkan hasil iterasi yang berbeda. Seperti yang telah dipaparkan, nilai *scalability* awal pada perbandingan proses bisnis Akademik *Modern* pada Gambar 3.7 dan proses

bisnis Akademik Mahasiswa pada Gambar 3.12 adalah 0.563700293, maka simulasi pertumbuhan dapat terjadi.

Berikut ditunjukkan hasil simulasi pertumbuhan dari Akademik *Salaf* sebagai *input*-an A dan Akademik Mahasiswa sebagai *input*-an B pada **Tabel 4.7** di bawah ini:

Tabel 4.7 Simulasi Pertumbuhan Akademik *Salaf* & Akademik Mahasiswa

PERCABANGAN					
Percobaan	Model B / Model A	Akademik Mahasiswa	Percobaan	Model B / Model A	Akademik Mahasiswa
	Akademik *Salaf*	0.208961765		Akademik *Salaf*	0.208961765
	Iterasi 1	-0.6047970503		Iterasi 1	-0.6047970503
SEQUENCE					
Percobaan	Model B / Model A	Akademik Mahasiswa	Percobaan	Model B / Model A	Akademik Mahasiswa
	Akademik *Salaf*	0.208961765		Akademik *Salaf*	0.208961765
	Iterasi 1	-0.227261843		Iterasi 1	-0.227261843

Menurut **Tabel 4.7** di atas, simulasi pertumbuhan percobaan pertama dan kedua pada pembobotan percabangan dan *sequence* mendapatkan hasil yang sama. Dilakukan juga simulasi pertumbuhan dengan pola AND, artinya percabangan yang lebih ke arah logika AND dan mendapatkan hasil iterasi yang berbeda. Seperti yang telah dipaparkan, nilai *scalability* awal pada perbandingan proses bisnis Akademik *Salaf* pada Gambar 3.2 dan proses bisnis Akademik Mahasiswa pada Gambar 3.12 adalah 0.208961765, maka simulasi pertumbuhan dapat terjadi tetapi tidak menghasilkan nilai minimum pada pembobotan percabangan maupun *sequence*.

2. Penerimaan Santri Baru

Dilakukan perhitungan perbandingan untuk menemukan nilai *scalability* pada bagian Penerimaan Santri Baru (PSB). Berikut diketahui nilai *scalability*

dari hasil perbandingan model PNML dari bagian PSB yang dipaparkan pada **Tabel 4.8** di bawah ini:

Tabel 4.8 Perbandingan Nilai *Scalability* PSB

Model B / Model A	**PSB *Salaf***	**PSB Mahasiswa**	**PSB *Tahfidz***	**PSB *Modern***
PSB *Salaf*	0.0	0.738986479	0.522002679	0.861318021
PSB Mahasiswa	-	0.0	0.648137494	0.774710141
PSB *Tahfidz*	-	-	0.0	0.763426248
PSB PSB *Modern*	-	-	-	0.0

Dari **Tabel 4.8** di atas, dapat dilihat didapatkan variasi nilai *scalability* dengan membandingkan dua model PNML dari bagian PSB. Jika *complexity input*-an A > *input*-an B, maka tidak dapat dilakukan simulasi pertumbuhan. Begitu juga sebaliknya, jika *complexity input*-an A < *input*-an B, maka dapat dilakukan simulasi pertumbuhan. Namun, jika pada tabel di atas terdapat kolom yang isinya tanda "-" artinya perbandingan antar dua model PNML menghasilkan nilai *scalability* tetapi tidak dapat dilakukan simulasi pertumbuhan. Jika membandingkan *input*-an A dan *input*-an B dengan model PNML yang sama, maka nilai *scalability* akan otomatis bernilai nol.

Berikut ditunjukkan hasil simulasi pertumbuhan dari PSB *Salaf* sebagai *input*-an A dan PSB Mahasiswa sebagai *input*-an B. Perbandingan antar model tersebut, *input*-an A mewakili proses bisnis yang *sequence*, sedangkan *input*-an B juga *sequence*. Hasil perbandingan ditunjukkan pada **Tabel 4.9** di bawah ini:

Tabel 4.9 Simulasi Pertumbuhan PSB *Salaf* & PSB Mahasiswa

PERCABANGAN					
Percobaan 1	Model B / Model A	PSB Mahasiswa	Percobaan 2	Model B / Model A	PSB Mahasiswa
	PSB *Salaf*	0.738986479		**PSB *Salaf***	0.738986479
	Iterasi 1	0.726220276		Iterasi 1	0.726220276
	Iterasi 2	0.714671456		Iterasi 2	0.714671456
	Iterasi 3	0.396798373		Iterasi 3	0.698399186
	Iterasi 4	0.360232698		Iterasi 4	0.360232698
	Iterasi 5	0.343893267		Iterasi 5	0.325136099
	Iterasi 6	0.104136463		Iterasi 6	-0.417385484
	Iterasi 7	-0.301030917		Iterasi 7	
AND	Iterasi 1	0.726220276	XOR	Iterasi 1	0.452440553
	Iterasi 2	0.698399186			-
	Iterasi 3	0.429342912			-
	Iterasi 4	0.360232698			-
	Iterasi 5	0.325136099			-
	Iterasi 6	0.291307257			-
	Iterasi 7	0.126606945			-
	Iterasi 8	-0.280194484			-
SEQUENCE					
Percobaan 1	Model B / Model A	PSB Mahasiswa	Percobaan 2	Model B / Model A	PSB Mahasiswa
	PSB *Salaf*	0.738986479		**PSB *Salaf***	0.738986479
	Iterasi 1	0.718399252		Iterasi 1	0.718399252
	Iterasi 2	0.692216161		Iterasi 2	0.692216161
	Iterasi 3	0.325136099		Iterasi 3	0.662568049
	Iterasi 4	0.269376531		Iterasi 4	0.269376531
	Iterasi 5	0.216432263		Iterasi 5	-0.175351604
	Iterasi 6	-0.096790891		Iterasi 6	

Menurut **Tabel 4.9** di atas, simulasi pertumbuhan percobaan pertama dan kedua pada pembobotan percabangan dan *sequence* mendapatkan hasil yang berbeda. Dilakukan juga simulasi pertumbuhan dengan pola AND, artinya percabangan yang lebih ke arah logika AND dan mendapatkan hasil iterasi yang berbeda. Seperti yang telah dipaparkan, nilai *scalability* awal pada perbandingan proses bisnis PSB *Salaf* pada Gambar 3.3 dan proses bisnis PSB Mahasiswa pada Gambar 3.13 adalah 0.738986479, maka simulasi pertumbuhan dapat terjadi.

Berikut ditunjukkan hasil simulasi pertumbuhan dari PSB *Salaf* sebagai *input*-an A dan PSB *Tahfidz* sebagai *input*-an B pada **Tabel 4.10** di bawah ini:

Tabel 4.10 Simulasi Pertumbuhan PSB *Salaf* & PSB *Tahfidz*

PERCABANGAN					
Percobaan 1	Model B / Model A	PSB *Tahfidz*	Percobaan 2	Model B / Model A	PSB *Tahfidz*
	PSB *Salaf*	0.522002679		**PSB *Salaf***	0.522002679
	Iterasi 1	0.499495927		Iterasi 1	0.499495927
	Iterasi 2	0.479160586		Iterasi 2	0.479160586
	Iterasi 3	0.460628268		Iterasi 3	-0.078743462
	Iterasi 4	0.438975908		Iterasi 4	
	Iterasi 5	-0.771977515		Iterasi 5	
SEQUENCE					
Percobaan 1	Model B / Model A	PSB *Tahfidz*	Percobaan 2	Model B / Model A	PSB *Tahfidz*
	PSB *Salaf*	0.522002679		**PSB *Salaf***	0.522002679
	Iterasi 1	0.485722515		Iterasi 1	0.485722515
	Iterasi 2	0.454800429		Iterasi 2	0.454800429
	Iterasi 3	0.409340828		Iterasi 3	-0.1813183436
	Iterasi 4	0.362354051		Iterasi 4	
	Iterasi 5	-0.364320554		Iterasi 5	

Menurut **Tabel 4.10** di atas, simulasi pertumbuhan percobaan pertama dan kedua pada pembobotan percabangan dan *sequence* mendapatkan hasil yang berbeda. Seperti yang telah dipaparkan, nilai *scalability* awal pada perbandingan proses bisnis PSB *Salaf* pada Gambar 3.3 dan proses bisnis PSB *Tahfidz* pada Gambar 3.18 adalah 0.522002679, maka simulasi pertumbuhan dapat terjadi.

Berikut ditunjukkan hasil simulasi pertumbuhan dari PSB *Salaf* sebagai *input*-an A dan PSB *Modern* sebagai *input*-an B pada **Tabel 4.11** di bawah ini:

Tabel 4.11 Simulasi Pertumbuhan PSB *Salaf* & PSB *Modern*

PERCABANGAN					
Percobaan 1	Model B / Model A	PSB *Modern*	Percobaan 2	Model B / Model A	PSB *Modern*
	PSB *Salaf*	0.861318021		**PSB *Salaf***	0.861318021
	Iterasi 1	0.854199439		Iterasi 1	0.854199439
	Iterasi 2	0.847644486		Iterasi 2	0.847644486
	Iterasi 3	0.841575858		Iterasi 3	0.841575858
	Iterasi 4	0.835930901		Iterasi 4	0.835930901
	Iterasi 5	0.830658030		Iterasi 5	0.491974092
	Iterasi 6	0.651428472		Iterasi 6	0.477103792
	Iterasi 7	0.527843539		Iterasi 7	0.370458052
	Iterasi 8	0.477103792		Iterasi 8	0.161294172
	Iterasi 9	0.466626466		Iterasi 9	-0.159571748
	Iterasi 10	0.399954774		Iterasi 10	

	Iterasi 11	0.266611391		Iterasi 11	
	Iterasi 12	0.055315809		Iterasi 12	
	Iterasi 13	-0.092424913		Iterasi 13	
	Iterasi 1	0.854199439		Iterasi 1	0.708398879
	Iterasi 2	0.847644486			
	Iterasi 3	0.841575858			
	Iterasi 4	0.828488576			
	Iterasi 5	0.670704506			
A	Iterasi 6	0.661316061	X		
N	Iterasi 7	0.651428472	O		
D	Iterasi 8	0.599969849	R		
	Iterasi 9	0.466626466			
	Iterasi 10	0.333283082			
	Iterasi 11	0.199939699			
	Iterasi 12	0.066596316			
	Iterasi 13	-0.133418758			

SEQUENCE

Percobaan 1	Model B / Model A	PSB *Modern*	Percobaan 2	Model B / Model A	PSB *Modern*
	PSB *Salaf*	0.861318021		**PSB *Salaf***	0.861318021
	Iterasi 1	0.849771960		Iterasi 1	0.849771960
	Iterasi 2	0.839649879		Iterasi 2	0.839649879
	Iterasi 3	0.322632123		Iterasi 3	0.830658030
	Iterasi 4	0.290332232		Iterasi 4	0.467749174
	Iterasi 5	-0.309416598		Iterasi 5	0.064702429
	Iterasi 6			Iterasi 6	-0.224233930

Menurut **Tabel 4.11** di atas, simulasi pertumbuhan percobaan pertama dan kedua pada pembobotan percabangan dan *sequence* mendapatkan hasil yang berbeda. Seperti yang telah dipaparkan, nilai *scalability* awal pada perbandingan proses bisnis PSB *Salaf* pada Gambar 3.3 dan proses bisnis PSB *Modern* pada Gambar 3.8 adalah 0.861318021, maka simulasi pertumbuhan dapat terjadi.

Berikut ditunjukkan hasil simulasi pertumbuhan dari PSB Mahasiswa sebagai *input*-an A dan PSB *Tahfidz* sebagai *input*-an B pada **Tabel 4.12** di bawah ini:

Tabel 4.12 Simulasi Pertumbuhan PSB Mahasiswa & PSB *Tahfidz*

PERCABANGAN					
Percobaan 1	Model B / Model A	PSB *Tahfidz*	Percobaan 2	Model B / Model A	PSB *Tahfidz*
	PSB Mahasiswa	0.648137494		PSB Mahasiswa	0.648137494
	Iterasi 1	0.635555362		Iterasi 1	0.635555362
	Iterasi 2	0.617608161		Iterasi 2	0.617608161
	Iterasi 3	0.597333255		Iterasi 3	0.597333255
	Iterasi 4	0.577793393		Iterasi 4	0.577793393
	Iterasi 5	0.298060054		Iterasi 5	0.199244047
	Iterasi 6	-0.823712599		Iterasi 6	-0.669966118
AND	Iterasi 1	0.635555362	XOR	Iterasi 1	-0.093333912
	Iterasi 2	0.617608161			
	Iterasi 3	0.597333255			
	Iterasi 4	0.155586787			
	Iterasi 5	-0.201133928			
SEQUENCE					
Percobaan 1	Model B / Model A	PSB *Tahfidz*	Percobaan 2	Model B / Model A	PSB *Tahfidz*
	PSB Mahasiswa	0.648137494		PSB Mahasiswa	0.648137494
	Iterasi 1	0.624548340		Iterasi 1	0.624548340
	Iterasi 2	0.181486344		Iterasi 2	0.590743172
	Iterasi 3	-1.205514677		Iterasi 3	0.117794128
	Iterasi 4			Iterasi 4	-0.885472380

Menurut **Tabel 4.12** di atas, simulasi pertumbuhan percobaan pertama dan kedua pada pembobotan percabangan dan *sequence* mendapatkan hasil yang berbeda. Seperti yang telah dipaparkan, nilai *scalability* awal pada perbandingan proses bisnis PSB Mahasiswa pada Gambar 3.13 dan proses bisnis PSB *Tahfidz* pada Gambar 3.18 adalah 0.648137494, maka simulasi pertumbuhan dapat terjadi.

Berikut ditunjukkan hasil simulasi pertumbuhan dari PSB Mahasiswa sebagai *input*-an A dan PSB *Modern* sebagai *input*-an B pada **Tabel 4.13** di bawah ini:

Tabel 4.13 Simulasi Pertumbuhan PSB Mahasiswa & PSB *Modern*

PERCABANGAN					
Percobaan 1	Model B / Model A	PSB *Modern*	Percobaan 2	Model B / Model A	PSB *Modern*
	PSB Mahasiswa	0.774710141		PSB Mahasiswa	0.7747101415
	Iterasi 1	0.766144866		Iterasi 1	0.766144866
	Iterasi 2	0.758176299		Iterasi 2	0.758176299
	Iterasi 3	0.750731732		Iterasi 3	0.750731732
	Iterasi 4	0.743750513		Iterasi 4	0.487501026
	Iterasi 5	0.519819976		Iterasi 5	0.039639952
	Iterasi 6	0.107142857		Iterasi 6	-0.484868520
	Iterasi 7	-0.395816626		Iterasi 7	
AND	Iterasi 1	0.766144866	XOR	Iterasi 1	0.298434600
	Iterasi 2	0.758176299		Iterasi 2	-0.450942202
	Iterasi 3	0.750731732			
	Iterasi 4	0.743750513			
	Iterasi 5	0.322057204			
	Iterasi 6	0.300437753			
	Iterasi 7	0.231488662			
	Iterasi 8	0.217166177			
	Iterasi 9	0.167011026			
	Iterasi 10	0.107142857			
	Iterasi 11	0.097515050			
	Iterasi 12	-0.128106186			
SEQUENCE					
Percobaan 1	Model B / Model A	PSB *Modern*	Percobaan 2	Model B / Model A	PSB *Modern*
	PSB Mahasiswa	0.774710141		PSB Mahasiswa	0.774710141
	Iterasi 1	0.760770952		Iterasi 1	0.760770952
	Iterasi 2	0.496711753		Iterasi 2	0.496711753
	Iterasi 3	0.474363040		Iterasi 3	0.211544560
	Iterasi 4	-0.118723338		Iterasi 4	-0.118723338

Menurut **Tabel 4.13** di atas, simulasi pertumbuhan percobaan pertama dan kedua pada pembobotan percabangan dan *sequence* mendapatkan hasil yang berbeda. Seperti yang telah dipaparkan, nilai *scalability* awal pada perbandingan proses bisnis PSB Mahasiswa pada Gambar 3.13 dan proses bisnis PSB *Modern* pada Gambar 3.8 adalah 0.774710141, maka simulasi pertumbuhan dapat terjadi.

Berikut ditunjukkan hasil simulasi pertumbuhan dari PSB *Tahfidz* sebagai *input*-an A dan PSB *Modern* sebagai *input*-an B pada **Tabel 4.14** di bawah ini:

Tabel 4.14 Simulasi Pertumbuhan PSB *Tahfidz* & PSB *Modern*

PERCABANGAN					
Percobaan 1	Model B / Model A	PSB *Modern*	Percobaan 2	Model B / Model A	PSB *Modern*
	PSB *Tahfidz*	0.763426248		**PSB *Tahfidz***	0.763426248
	Iterasi 1	0.755639770		Iterasi 1	0.755639770
	Iterasi 2	0.496711753		Iterasi 2	0.496711753
	Iterasi 3	0.483034668		Iterasi 3	0.483034668
	Iterasi 4	0.291710389		Iterasi 4	0.366774218
	Iterasi 5	-0.284543578		Iterasi 5	0.291710389
	Iterasi 6			Iterasi 6	-0.468049804
AND	Iterasi 1	0.755639770	XOR	Iterasi 1	0.266919312
	Iterasi 2	0.748355876		Iterasi 2	-0.006576492
	Iterasi 3	0.683111932			
	Iterasi 4	0.645855194			
	Iterasi 5	0.429292311			
	Iterasi 6	0.361790106			
	Iterasi 7	0.270617264			
	Iterasi 8	0.088271580			
	Iterasi 9	-0.094074103			
SEQUENCE					
Percobaan 1	Model B / Model A	PSB *Modern*	Percobaan 2	Model B / Model A	PSB *Modern*
	PSB *Tahfidz*	0.763426248		**PSB *Tahfidz***	0.763426248
	Iterasi 1	0.501463465		Iterasi 1	0.501463465
	Iterasi 2	0.217984614		Iterasi 2	0.217984614
	Iterasi 3	-0.3802313362		Iterasi 3	-0.1041850689

Menurut **Tabel 4.14** di atas, simulasi pertumbuhan percobaan pertama dan kedua pada pembobotan percabangan dan *sequence* mendapatkan hasil iterasi yang berbeda. Namun, didapatkan nilai *scalability* minimum yang sama pada percobaan pertama dan kedua di masing-masing pembobotan. Seperti yang telah dipaparkan, nilai *scalability* awal pada perbandingan proses bisnis PSB *Tahfidz* pada Gambar 3.18 dan proses bisnis PSB *Modern* pada Gambar 3.8 adalah 0.763426248, maka simulasi pertumbuhan dapat terjadi.

3. Kesantrian

Dilakukan perhitungan perbandingan untuk menemukan nilai *scalability* pada bagian Kesantrian. Berikut diketahui beberapa nilai *scalability* yang

didapatkan dari perbandingan model PNML bagian Kesantrian yang dipaparkan pada **Tabel 4.15** di bawah ini:

Tabel 4.15 Perbandingan Nilai *Scalability* Kesantrian

Model B / Model A	**Kesantrian *Salaf***	**Kesantrian *Modern***	**Kesantrian Mahasiswa**	**Kesantrian *Tahfidz***
Kesantrian *Salaf*	0.0	0.796102695	0.833123626	0.872625042
Kesantrian *Modern*	-	0.0	0.774498119	0.793487116
Kesantrian Mahasiswa	-	-	0.0	0.564452549
Kesantrian *Tahfidz*	-	-	-	0.0

Dari **Tabel 4.15** di atas, dapat dilihat didapatkan variasi nilai *scalability* dengan membandingkan dua model PNML dari bagian Kesantrian. Jika *complexity input*-an A > *input*-an B, maka tidak dapat dilakukan simulasi pertumbuhan. Begitu juga sebaliknya, jika *complexity input*-an A < *input*-an B, maka dapat dilakukan simulasi pertumbuhan. Namun, jika pada tabel di atas terdapat kolom yang isinya tanda "-" artinya perbandingan antar dua model PNML menghasilkan nilai *scalability* tetapi tidak dapat dilakukan simulasi pertumbuhan. Jika membandingkan *input*-an A dan *input*-an B dengan model PNML yang sama, maka nilai *scalability* akan otomatis bernilai nol.

Berikut ditunjukkan hasil simulasi pertumbuhan dari Kesantrian *Salaf* sebagai *input*-an A dan Kesantrian *Modern* sebagai *input*-an B. Perbandingan antar model tersebut, *input*-an A mewakili proses bisnis yang *sequence*, sedangkan *input*-an B mewakili proses bisnis bercabang (level1). Hasil perbandingan ditunjukkan pada **Tabel 4.16** di bawah ini:

Tabel 4.16 Simulasi Pertumbuhan Kesantrian *Salaf* & Kesantrian *Modern*

PERCABANGAN					
Percobaan 1	Model B / Model A	Kesantrian *Modern*	Percobaan 2	Model B / Model A	Kesantrian *Modern*
	Kesantrian *Salaf*	0.796102695		Kesantrian *Salaf*	0.796102695
	Iterasi 1	0.788102159		Iterasi 1	0.788102159
	Iterasi 2	0.549165580		Iterasi 2	0.549165580
	Iterasi 3	0.498459693		Iterasi 3	0.003810828
	Iterasi 4	0.474481575		Iterasi 4	-0.817045951
	Iterasi 5	0.063324737		Iterasi 5	
	Iterasi 6	-0.705645115		Iterasi 6	
AND	Iterasi 1	0.788102159	XOR	Iterasi 1	0.576204 3189
	Iterasi 2	0.774582790		Iterasi 2	0.098331 1614
	Iterasi 3	0.523308131			
	Iterasi 4	0.498459693			
	Iterasi 5	0.474481575			
	Iterasi 6	0.063324737			
	Iterasi 7	-0.527059460			
SEQUENCE					
Percobaan 1	Model B / Model A	Kesantrian *Modern*	Percobaan 2	Model B / Model A	Kesantrian *Modern*
	Kesantrian *Salaf*	0.796102695		Kesantrian *Salaf*	0.796102695
	Iterasi 1	0.779018784		Iterasi 1	0.779018784
	Iterasi 2	0.757460329		Iterasi 2	0.757460329
	Iterasi 3	0.474481575		Iterasi 3	0.514920658
	Iterasi 4	0.154230861		Iterasi 4	0.029841316
	Iterasi 5	-0.483168818		Iterasi 5	-0.576555274

Menurut **Tabel 4.16** di atas, simulasi pertumbuhan percobaan pertama dan kedua pada pembobotan percabangan dan *sequence* mendapatkan hasil iterasi yang berbeda. Seperti yang telah dipaparkan, nilai *scalability* awal pada perbandingan proses bisnis Kesantrian *Salaf* pada Gambar 3.4 dan proses bisnis Kesantrian *Modern* pada Gambar 3.9 adalah 0.796102695, maka simulasi pertumbuhan dapat terjadi.

Berikut ditunjukkan hasil simulasi pertumbuhan dari Kesantrian *Salaf* sebagai *input*-an A dan Kesantrian Mahasiswa sebagai *input*-an B. Perbandingan antar model tersebut, *input*-an A mewakili proses bisnis yang

sequence, sedangkan *input*-an B mewakili proses bisnis bercabang (level2). Hasil perbandingan ditunjukkan pada **Tabel 4.17** di bawah ini:

Tabel 4.17 Simulasi Pertumbuhan Kesantrian *Salaf* & Kesantrian Mahasiswa

PERCABANGAN					
	Model B / Model A	Kesantrian Mahasiswa		Model B / Model A	Kesantrian Mahasiswa
Percobaan 1	Kesantrian *Salaf*	0.833123626	Percobaan 2	Kesantrian *Salaf*	0.833123626
	Iterasi 1	0.826655098		Iterasi 1	0.826655098
	Iterasi 2	0.641495601		Iterasi 2	0.641495601
	Iterasi 3	0.402765794		Iterasi 3	0.431896459
	Iterasi 4	0.374685255		Iterasi 4	0.004609657
	Iterasi 5	0.192134585		Iterasi 5	-0.459067736
	Iteasi 6	-0.051843990		Iterasi 6	
AND	Iterasi 1	0.826655098	XOR	Iterasi 1	0.653310196
	Iterasi 2	0.810632153			
	Iterasi 3	0.800921931			
	Iterasi 4	0.791561751			
	Iterasi 5	0.610338264			
	Iteasi 6	0.424445281			
	Iterasi 7	0.084779909			
	Iterasi 8	-0.215398152			
SEQUENCE					
	Model B / Model A	Kesantrian Mahasiswa		Model B / Model A	Kesantrian Mahasiswa
Percobaan 1	Kesantrian *Salaf*	0.833123626	Percobaan 2	Kesantrian *Salaf*	0.833123626
	Iterasi 1	0.822660335		Iterasi 1	0.822660335
	Iterasi 2	0.807353341		Iterasi 2	0.614706682
	Iterasi 3	0.583123503		Iterasi 3	0.438610349
	Iterasi 4	-0.116898667		Iterasi 4	0.328689739
	Iterasi 5			Iterasi 5	0.281690375

Menurut **Tabel 4.17** di atas, simulasi pertumbuhan percobaan pertama dan kedua pada pembobotan percabangan dan *sequence* mendapatkan hasil iterasi yang berbeda. Seperti yang telah dipaparkan, nilai *scalability* awal pada perbandingan proses bisnis Kesantrian *Salaf* pada Gambar 3.4 dan proses bisnis Kesantrian Mahasiswa pada Gambar 3.14 adalah 0.833123626, maka simulasi pertumbuhan dapat terjadi.

Berikut ditunjukkan hasil simulasi pertumbuhan dari Kesantrian *Salaf* sebagai *input*-an A dan Kesantrian *Tahfidz* sebagai *input*-an B pada **Tabel 4.18** di bawah ini:

Tabel 4.18 Simulasi Pertumbuhan Kesantrian *Salaf* & Kesantrian *Tahfidz*

PERCABANGAN					
Percobaan 1	**Model B / Model A**	**Kesantrian *Tahfidz***	Percobaan 2	**Model B / Model A**	**Kesantrian *Tahfidz***
	Kesantrian *Salaf*	0.872625042		**Kesantrian *Salaf***	0.872625042
	Iterasi 1	0.867386789		Iterasi 1	0.867386789
	Iterasi 2	0.725140377		Iterasi 2	0.862570188
	Iterasi 3	0.716231261		Iterasi 3	0.707949818
	Iterasi 4	0.707949818		Iterasi 4	0.706278782
	Iterasi 5	0.245337601		Iterasi 5	0.573116621
	Iterasi 6	-0.103762111		Iterasi 6	0.313752347
	Iterasi 7			Iterasi 7	0.001692486
	Iterasi 8			Iterasi 8	-0.330530568
AND	Iterasi 1	0.867386789	XOR	Iterasi 1	0.734773578
	Iterasi 2	0.862570188		Iterasi 2	0.312850942
	Iterasi 3	0.716231261		Iterasi 3	0.148693784
	Iterasi 4	0.412557565		Iterasi 4	0.123849455
	Iterasi 5	0.269874546			
	Iterasi 6	0.245337601			
	Iterasi 7	0.039253286			
	Iterasi 8	0.014095054			
	Iterasi 9	-0.030547344			
SEQUENCE					
Percobaan 1	**Model B / Model A**	**Kesantrian *Tahfidz***	Percobaan 2	**Model B / Model A**	**Kesantrian *Tahfidz***
	Kesantrian *Salaf*	0.872625042		**Kesantrian *Salaf***	0.872625042
	Iterasi 1	0.864132817		Iterasi 1	0.864132817
	Iterasi 2	0.713405257		Iterasi 2	0.713405257
	Iterasi 4	0.547202561		Iterasi 4	0.547202561
	Iterasi 5	0.027649489		Iterasi 5	0.513824744
	Iterasi 6	0.022859146		Iterasi 6	0.355239431
	Iterasi 7	-0.034857108		Iterasi 7	-0.034857108

Menurut **Tabel 4.18** di atas, simulasi pertumbuhan percobaan pertama dan kedua pada pembobotan percabangan dan *sequence* mendapatkan hasil iterasi yang berbeda. Seperti yang telah dipaparkan, nilai *scalability* awal pada perbandingan proses bisnis Kesantrian *Salaf* pada Gambar 3.4 dan proses bisnis

Kesantrian *Tahfidz* pada Gambar 3.19 adalah 0.872625042, maka simulasi pertumbuhan dapat terjadi. Batas nilai *scalability*: 0.5.

Berikut ditunjukkan hasil simulasi pertumbuhan dari Kesantrian *Modern* sebagai *input*-an A dan Kesantrian Mahasiswa sebagai *input*-an B. Perbandingan antar model tersebut, *input*-an A mewakili proses bisnis yang bercabang (level1), sedangkan *input*-an B mewakili proses bisnis bercabang (level2). Hasil perbandingan pada **Tabel 4.19** di bawah ini:

Tabel 4.19 Simulasi Pertumbuhan Kesantrian *Modern* & Kesantrian Mahasiswa

PERCABANGAN					
	Model B / Model A	Kesantrian Mahasiswa		Model B / Model A	Kesantrian Mahasiswa
Percobaan 1	**Kesantrian *Modern***	0.774498119	Percobaan 2	**Kesantrian *Modern***	0.774498119
	Iterasi 1	0.649677419		Iterasi 1	0.766451612
	Iterasi 2	0.540030020		Iterasi 2	0.752793074
	Iterasi 3	0.372788215		Iterasi 3	0.463843427
	Iterasi 4	0.252213302		Iterasi 4	0.126176898
	Iterasi 5	0.043812340		Iterasi 5	-0.296155308
	Iterasi 6	-0.063479899		Iterasi 6	
AND	Iterasi 1	0.727034005	XOR	Iterasi 1	0.649677419
	Iterasi 2	0.714802706			
	Iterasi 3	0.655022515			
	Iterasi 4	0.626106651			
	Iterasi 5	0.514591091			
	Iterasi 6	0.476762125			
	Iterasi 7	0.396263991			
	Iterasi 8	0.315765856			
	Iterasi 9	0.275516789			
	Iterasi 10	0.235267722			
	Iterasi 11	0.154769587			
	Iterasi 12	0.114520520			
	Iterasi 13	0.034022385			
	Iterasi 14	-0.006226681			
SEQUENCE					
	Model B / Model A	Kesantrian Mahasiswa		Model B / Model A	Kesantrian Mahasiswa
Percobaan 1	**Kesantrian *Modern***	0.774498119	Percobaan 2	**Kesantrian *Modern***	0.774498119
	Iterasi 1	0.635922378		Iterasi 1	0.757281568
	Iterasi 2	0.603120919		Iterasi 2	0.603120919
	Iterasi 3			Iterasi 3	

Menurut **Tabel 4.19** di atas, simulasi pertumbuhan percobaan pertama dan kedua pada pembobotan percabangan dan *sequence* mendapatkan hasil iterasi yang berbeda. Seperti yang telah dipaparkan, nilai *scalability* awal pada perbandingan proses bisnis Kesantrian *Modern* pada Gambar 3.9 dan proses bisnis Kesantrian Mahasiswa pada Gambar 3.14 adalah 0.774498119, maka simulasi pertumbuhan dapat terjadi.

Berikut ditunjukkan hasil simulasi pertumbuhan dari Kesantrian *Modern* sebagai *input*-an A dan Kesantrian *Tahfidz* sebagai *input*-an B pada **Tabel 4.20** di bawah ini:

Tabel 4.20 Simulasi Pertumbuhan Kesantrian *Modern* & Kesantrian *Tahfidz*

PERCABANGAN					
Percobaan 1	Model B / Model A	Kesantrian *Tahfidz*	Percobaan 2	Model B / Model A	Kesantrian *Tahfidz*
	Kesantrian *Modern*	0.793487116		Kesantrian *Modern*	0.793487116
	Iterasi 1	0.764356498		Iterasi 1	0.785839328
	Iterasi 2	0.753643194		Iterasi 2	0.778755557
	Iterasi 3	0.528271294		Iterasi 3	0.772162620
	Iterasi 4	0.502868278		Iterasi 4	0.646534748
	Iterasi 5	0.445738424		Iterasi 5	0.260929584
	Iterasi 6	0.404048323		Iterasi 6	
	Iterasi 7	0.397744599		Iterasi 7	
	Iterasi 8	0.351417261		Iterasi 8	
	Iterasi 9	0.258762584		Iterasi 9	
	Iterasi 10	0.109681745		Iterasi 10	
	Iterasi 11	0.004887985		Iterasi 11	
	Iterasi 12	-0.054014626		Iterasi 12	
SEQUENCE					
Percobaan 1	Model B / Model A	Kesantrian *Tahfidz*	Percobaan 2	Model B / Model A	Kesantrian *Tahfidz*
	Kesantrian *Modern*	0.793487116		Kesantrian *Modern*	0.793487116
	Iterasi 1	0.671588437		Iterasi 1	0.781058958
	Iterasi 2	0.630464792		Iterasi 2	0.671588437
	Iterasi 3	0.152192670		Iterasi 3	0.655094957
	Iterasi 4	0.108865473		Iterasi 4	0.152192670
	Iterasi 5			Iterasi 5	

Menurut Tabel **Tabel 4.20** di atas, simulasi pertumbuhan percobaan pertama dan kedua pada pembobotan percabangan dan *sequence* mendapatkan

hasil iterasi yang berbeda. Seperti yang telah dipaparkan, nilai *scalability* awal pada perbandingan proses bisnis Kesantrian *Modern* pada Gambar 3.9 dan proses bisnis Kesantrian *Tahfidz* pada Gambar 3.19 adalah 0.793487116, maka simulasi pertumbuhan dapat terjadi.

Berikut ditunjukkan hasil simulasi pertumbuhan dari Kesantrian Mahasiswa sebagai *input*-an A dan Kesantrian *Tahfidz* sebagai *input*-an B. Perbandingan antar model tersebut, *input*-an A mewakili proses bisnis yang bercabang (level2), sedangkan *input*-an B mewakili proses bisnis bercabang (level2). Hasil perbandingan pada **Tabel 4.21** di bawah ini:

Tabel 4.21 Simulasi Pertumbuhan Kesantrian Mahasiswa & Kesantrian *Tahfidz*

PERCABANGAN					
Percobaan 1	Model B / Model A	Kesantrian *Tahfidz*	Percobaan 2	Model B / Model A	Kesantrian *Tahfidz*
	Kesantrian Mahasiswa	0.564452549		**Kesantrian Mahasiswa**	0.564452549
	Iterasi 1	0.334201110		Iterasi 1	0.513824744
	Iterasi 2	0.351766326		Iterasi 2	0.528410240
	Iterasi 3	0.056820481		Iterasi 3	0.300447032
	Iterasi 4	-0.468436252		Iterasi 4	0.209153134
	Iterasi 5			Iterasi 4	0.067008307
	Iterasi 6			Iterasi 5	-0.277288338
AND	Iterasi 1	0.521922471	XOR		-
	Iterasi 2	0.415554382			-
	Iterasi 3	0.392491989			-
	Iterasi 4	0.351766326			-
	Iterasi 5	0.020625845			-
	Iterasi 6	-0.054462487			-
					-
					-
SEQUENCE					
Percobaan 1	Model B / Model A	Kesantrian *Tahfidz*	Percobaan 2	Model B / Model A	Kesantrian *Tahfidz*
	Kesantrian Mahasiswa	0.564452549		**Kesantrian Mahasiswa**	0.564452549
	Iterasi 1	0.480054563		Iterasi 1	0.480054563
	Iterasi 2	0.363071680		Iterasi 2	0.132089580
	Iterasi 3	-0.012562156		Iterasi 3	-0.009939222

Menurut **Tabel 4.21** di atas, simulasi pertumbuhan percobaan pertama dan kedua pada pembobotan percabangan dan *sequence* mendapatkan hasil iterasi

yang berbeda. Seperti yang telah dipaparkan, nilai *scalability* awal pada perbandingan proses bisnis Kesantrian Mahasiswa pada Gambar 3.14 dan proses bisnis Kesantrian *Tahfidz* pada Gambar 3.19 adalah 0.564452549, maka simulasi pertumbuhan dapat terjadi.

4. Sarana dan Prasarana

Dilakukan perhitungan perbandingan untuk menemukan nilai *scalability* pada bagian Sarana dan Prasarana. Berikut didapatkan beberapa nilai *scalability* dengan membandingkan model PNML dari bagian Sarana dan Prasarana (Sarpras). Ditunjukkan pada **Tabel 4.22** di bawah ini:

Tabel 4.22 Perbandingan Nilai *Scalability* Sarpras

Model B / Model A	Sarpras *Tahfidz*	Sarpras *Modern*	Sarpras Mahasiswa	Sarpras *Salaf*
Sarpras *Tahfidz*	0.0	0.743212916	0.693283206	0.697486098
Sarpras *Modern*	-	0.0	0.421982758	0.500643627
Sarpras Mahasiswa	-	-	0.0	0.500643627
Sarpras *Salaf*	-	-	-	0.0

Dari **Tabel 4.22** di atas, dapat dilihat didapatkan variasi nilai *scalability* dengan membandingkan dua model PNML dari bagian Sarpras. Jika *complexity input*-an A > *input*-an B, maka tidak dapat dilakukan simulasi pertumbuhan. Begitu juga sebaliknya, jika *complexity input*-an A < *input*-an B, maka dapat dilakukan simulasi pertumbuhan. Namun, jika pada tabel di atas terdapat kolom yang isinya tanda "-" artinya perbandingan antar dua model PNML menghasilkan nilai *scalability* tetapi tidak dapat dilakukan simulasi pertumbuhan. Jika membandingkan *input*-an A dan *input*-an B dengan model PNML yang sama, maka nilai *scalability* akan otomatis bernilai nol.

Berikut ditunjukkan hasil simulasi pertumbuhan dari Sarpras *Tahfidz* sebagai *input*-an A dan Sarpras *Modern* sebagai *input*-an B pada **Tabel 4.23** di bawah ini:

Tabel 4.23 Simulasi Pertumbuhan Sarpras *Tahfidz* & Sarpras *Modern*

PERCABANGAN					
	Model B / Model A	Sarpras *Modern*		Model B / Model A	Sarpras *Modern*
Percobaan 1	Sarpras *Tahfidz*	0.743212916	Percobaan 2	Sarpras *Tahfidz*	0.743212916
	Iterasi 1	0.722678554		Iterasi 1	0.722678554
	Iterasi 2	0.704649298		Iterasi 2	0.704649298
	Iteras 3	0.377222968		Iteras 3	0.688611484
	Iterasi 4	0.348383144		Iterasi 4	0.674191572
	Iterasi 5	-0.355563819		Iterasi 5	0.661109045
	Iterasi 6			Iterasi 6	0.642608122
	Iterasi 7			Iterasi 7	0.621789123
	Iterasi 8			Iterasi 8	0.203573048
	Iterasi 9			Iterasi 9	-0.670037541
SEQUENCE					
Percobaan 1	Model B / Model A	Sarpras *Modern*	Percobaan 2	Model B / Model A	Sarpras *Modern*
	Sarpras *Tahfidz*	0.743212916		Sarpras *Tahfidz*	0.743212916
	Iterasi 1	0.710415892		Iterasi 1	0.710415892
	Iterasi 2	0.683640516		Iterasi 2	0.050921550
	Iteras 3	0.661109045		Iteras 3	-0.355563819
	Iterasi 4	0.257263975		Iterasi 4	
	Iterasi 5	-0.618876659		Iterasi 5	

Menurut **Tabel 4.23** di atas, simulasi pertumbuhan percobaan pertama dan kedua pada pembobotan percabangan dan *sequence* mendapatkan hasil iterasi yang berbeda. Seperti yang telah dipaparkan, nilai *scalability* awal pada perbandingan proses bisnis Sarpras *Tahfidz* pada Gambar 3.20 dan proses bisnis Sarpras *Modern* pada Gambar 3.10 adalah 0.743212916, maka simulasi pertumbuhan dapat terjadi.

Berikut ditunjukkan hasil simulasi pertumbuhan dari Sarpras *Tahfidz* sebagai *input*-an A dan Sarpras Mahasiswa sebagai *input*-an B pada **Tabel 4.24** di bawah ini:

Tabel 4.24 Simulasi Pertumbuhan Sarpras *Tahfidz* & Sarpras Mahasiswa

PERCABANGAN					
Percobaan 1	Model B / Model A	Sarpras Mahasiswa	Percobaan 2	Model B / Model A	Sarpras Mahasiswa
	Sarpras *Tahfidz*	0.693283206		**Kesantrian *Tahfidz***	0.693283206
	Iterasi 1	0.668960327		Iterasi 1	0.668960327
	Iterasi 2	0.647660661		Iterasi 2	0.647660661
	Iterasi 3	0.257502577		Iterasi 3	0.628751288
	Iterasi 4	-0.552898734		Iterasi 4	-0.164674050
SEQUENCE					
Percobaan 1	Model B / Model A	Sarpras Mahasiswa	Percobaan 2	Model B / Model A	Sarpras Mahasiswa
	Sarpras *Tahfidz*	0.693283206		**Kesantrian *Tahfidz***	0.693283206
	Iterasi 1	0.654468090		Iterasi 1	0.654468090
	Iterasi 2	0.622896612		Iterasi 2	0.622896612
	Iterasi 3	0.192783570		Iterasi 3	0.596391785
	Iterasi 4	-0.767397093		Iterasi 4	-0.767397093

Menurut **Tabel 4.24** di atas, simulasi pertumbuhan percobaan pertama dan kedua pada pembobotan percabangan dan *sequence* mendapatkan hasil iterasi yang berbeda. Seperti yang telah dipaparkan, nilai *scalability* awal pada perbandingan proses bisnis Sarpras *Tahfidz* pada Gambar 3.20 dan proses bisnis Sarpras Mahasiswa pada Gambar 3.15 adalah 0.693283206, maka simulasi pertumbuhan dapat terjadi.

Berikut ditunjukkan hasil simulasi pertumbuhan dari Sarpras *Tahfidz* sebagai *input*-an A dan Sarpras *Salaf* sebagai *input*-an B pada **Tabel 4.25** di bawah ini:

Tabel 4.25 Simulasi Pertumbuhan Sarpras *Tahfidz* & Sarpras *Salaf*

PERCABANGAN					
Percobaan 1	Model B / Model A	Sarpras *Salaf*	Percobaan 2	Model B / Model A	Sarpras *Salaf*
	Sarpras *Tahfidz*	0.697486098		**Sarpras *Tahfidz***	0.697486098
	Iterasi 1	0.676711309		Iterasi 1	0.676711309
	Iterasi 2	0.642963573		Iterasi 2	0.285927147
	Iterasi 3	0.611535559		Iterasi 3	-0.553857762
	Iterasi 4	0.163768737		Iterasi 4	
	Iterasi 5	-0.785467673		Iterasi 5	

SEQUENCE					
Percobaan 1	Model B / Model A	Sarpras *Salaf*	Percobaan 2	Model B / Model A	Sarpras *Salaf*
	Sarpras *Tahfidz*	0.697486098		**Sarpras *Tahfidz***	0.697486098
	Iterasi 1	0.653921747		Iterasi 1	
	Iterasi 2	0.202951407		Iterasi 2	0.601475703
	Iterasi 3	-0.339100754		Iterasi 3	0.553633081
	Iterasi 4			Iterasi 4	0.017906165
	Iterasi 5			Iterasi 5	-1.667169930

Menurut **Tabel 4.25** di atas, simulasi pertumbuhan percobaan pertama dan kedua pada pembobotan percabangan dan *sequence* mendapatkan hasil iterasi yang berbeda. Seperti yang telah dipaparkan, nilai *scalability* awal pada perbandingan proses bisnis Sarpras *Tahfidz* pada Gambar 3.20 dan proses bisnis Sarpras *Salaf* pada Gambar 3.5 adalah 0.697486098, maka simulasi pertumbuhan dapat terjadi.

Berikut ditunjukkan hasil simulasi pertumbuhan dari Sarpras *Modern* sebagai *input*-an A dan Sarpras Mahasiswa sebagai *input*-an B pada **Tabel 4.26** di bawah ini:

Tabel 4.26 Simulasi Pertumbuhan Sarpras *Modern* & Sarpras Mahasiswa

PERCABANGAN					
Percobaan 1	Model B / Model A	Sarpras Mahasiswa	Percobaan 2	Model B / Model A	Sarpras Mahasiswa
	Sarpras *Modern*	0.421982758		**Sarpras *Modern***	0.421982758
	Iterasi 1	0.387493189		Iterasi 1	0.387493189
	Iterasi 2	0.354466534		Iterasi 2	0.354466534
	Iterasi 3	0.322690051		Iterasi 3	-0.354619896
	Iterasi 4	0.291992727		Iterasi 4	
	Iterasi 5	-1.951059255		Iterasi 5	
SEQUENCE					
Percobaan 1	Model B / Model A	Sarpras Mahasiswa	Percobaan 2	Model B / Model A	Sarpras Mahasiswa
	Sarpras *Modern*	0.421982758		**Sarpras *Modern***	0.421982758
	Iterasi 1	0.365326980		Iterasi 1	0.365326980
	Iterasi 2	0.312345158		Iterasi 2	-0.375309683
	Iterasi 3	-0.475529627		Iterasi 3	

Menurut **Tabel 4.26** di atas, simulasi pertumbuhan percobaan pertama dan kedua pada pembobotan percabangan dan *sequence* mendapatkan hasil iterasi yang berbeda. Seperti yang telah dipaparkan, nilai *scalability* awal pada perbandingan proses bisnis Sarpras *Modern* pada Gambar 3.10 dan proses bisnis Sarpras Mahasiswa pada Gambar 3.15 adalah 0.421982758, maka simulasi pertumbuhan dapat terjadi.

Berikut ditunjukkan hasil simulasi pertumbuhan dari Sarpras *Modern* sebagai *input*-an A dan Sarpras *Salaf* sebagai *input*-an B pada **Tabel 4.27** di bawah ini:

Tabel 4.27 Simulasi Pertumbuhan Sarpras *Modern* & Sarpras *Salaf*

PERCABANGAN					
Percobaan 1	Model B / Model A	Sarpras *Salaf*	Percobaan 2	Model B / Model A	Sarpras *Salaf*
	Sarpras *Modern*	0.500643627		Sarpras *Modern*	0.500643627
	Iterasi 1	0.471584145		Iterasi 1	0.471584145
	Iterasi 2	0.443452669		Iterasi 2	0.443452669
	Iterasi 3	-0.167801152		Iterasi 3	0.416099423
	Iterasi 4			Iterasi 4	0.389407312
	Iterasi 5			Iterasi 5	0.371933090
	Iterasi 6			Iterasi 6	-0.935936046
SEQUENCE					
Percobaan 1	Model B / Model A	Sarpras *Salaf*	Percobaan 2	Model B / Model A	Sarpras *Salaf*
	Sarpras *Modern*	0.500643627		Sarpras *Modern*	0.500643627
	Iterasi 1	0.452736736		Iterasi 1	0.452736736
	Iterasi 2	0.407133619		Iterasi 2	0.407133619
	Iterasi 3	-0.273433574		Iterasi 3	0.363283212
	Iterasi 4			Iterasi 4	-0.226794109

Menurut **Tabel 4.27** di atas, simulasi pertumbuhan percobaan pertama dan kedua pada pembobotan percabangan dan *sequence* mendapatkan hasil iterasi yang berbeda. Seperti yang telah dipaparkan, nilai *scalability* awal pada perbandingan proses bisnis Sarpras *Modern* pada Gambar 3.10 dan proses

bisnis Sarpras *Salaf* pada Gambar 3.5 adalah 0.500643627, maka simulasi pertumbuhan dapat terjadi.

Berikut ditunjukkan hasil simulasi pertumbuhan dari Sarpras Mahasiswa sebagai *input*-an A dan Sarpras *Salaf* sebagai *input*-an B pada **Tabel 4.28** di bawah ini:

Tabel 4.28 Simulasi Pertumbuhan Sarpras Mahasiswa & Sarpras *Salaf*

PERCABANGAN					
Percobaan 1	Model B / Model A	Sarpras *Salaf*	Percobaan 2	Model B / Model A	Sarpras *Salaf*
	Sarpras Mahasiswa	0.500643627		Sarpras Mahasiswa	0.500643627
	Iterasi 1	0.471584145		Iterasi 1	0.471584145
	Iterasi 2	0.443452669		Iterasi 2	0.443452669
	Iterasi 3	0.416099423		Iterasi 3	-0.167801152
	Iterasi 4	0.389407312		Iterasi 4	
	Iterasi 5	-0.256133818		Iterasi 5	
SEQUENCE					
Percobaan 1	Model B / Model A	Sarpras *Salaf*	Percobaan 2	Model B / Model A	Sarpras *Salaf*
	Sarpras Mahasiswa	0.500643627		Sarpras Mahasiswa	0.500643627
	Iterasi 1	0.452736736		Iterasi 1	0.452736736
	Iterasi 2	0.407133619		Iterasi 2	-0.185732761
	Iterasi 3	-0.273433574		Iterasi 3	

Menurut **Tabel 4.28** di atas, simulasi pertumbuhan percobaan pertama dan kedua pada pembobotan percabangan dan *sequence* mendapatkan hasil iterasi yang berbeda. Seperti yang telah dipaparkan, nilai *scalability* awal pada perbandingan proses bisnis Sarpras Mahasiswa pada Gambar 3.15 dan proses bisnis Sarpras *Salaf* pada Gambar 3.5 adalah 0.500643627, maka simulasi pertumbuhan dapat terjadi.

5. Kepegawaian

Dilakukan perhitungan perbandingan untuk menemukan nilai *scalability* pada bagian Kepegawaian. Berikut didapatkan beberapa nilai *scalability* dari

perbandingan model PNML bagian Kepegawaian yang dipaparkan pada **Tabel 4.29** di bawah ini:

Tabel 4.29 Perbandingan Nilai *Scalability* Kepegawaian

Model A \ Model B	**Kepegawaian *Salaf***	**Kepegawaian Mahasiswa**	**Kepegawaian *Tahfidz***	**Kepegawaian *Modern***
Kepegawaian *Salaf*	0.0	0.779933590	0.973988914	0.973939673
Kepegawaian Mahasiswa	-	0.0	0.889639114	0.936562248
Kepegawaian *Tahfidz*	-	-	0.0	0.586118495
Kepegawaian *Modern*	-	-	-	0.0

Dari **Tabel 4.29** di atas, dapat dilihat didapatkan variasi nilai *scalability* dengan membandingkan dua model PNML dari bagian Kepegawaian. Jika *complexity input*-an A > *input*-an B, maka tidak dapat dilakukan simulasi pertumbuhan. Begitu juga sebaliknya, jika *complexity input*-an A < *input*-an B, maka dapat dilakukan simulasi pertumbuhan. Namun, jika pada tabel di atas terdapat kolom yang isinya tanda "-" artinya perbandingan antar dua model PNML menghasilkan nilai *scalability* tetapi tidak dapat dilakukan simulasi pertumbuhan. Jika membandingkan *input*-an A dan *input*-an B dengan model PNML yang sama, maka nilai *scalability* akan otomatis bernilai nol.

Berikut ditunjukkan hasil simulasi pertumbuhan dari Kepegawaian *Salaf* sebagai *input*-an A dan Kepegawaian Mahasiswa sebagai *input*-an B. Perbandingan antar model tersebut, *input*-an A mewakili proses bisnis yang *sequence*, sedangkan *input*-an B juga mewakili proses bisnis *sequence*. Hasil perbandingan pada **Tabel 4.30** di bawah ini:

Tabel 4.30 Simulasi Pertumbuhan Kepegawaian *Salaf* & Kepegawaian Mahasiswa

PERCABANGAN					
Percobaan 1	Model B / Model A	Kepegawaian Mahasiswa	Percobaan 2	Model B / Model A	Kepegawaian Mahasiswa
	Kepegawaian *Salaf*	0.779933590		Kepegawaian *Salaf*	0.779933590
	Iterasi 1	0.769054388		Iterasi 1	0.769054388
	Iterasi 2	0.759194000		Iterasi 2	0.759194000
	Iterasi 4	0.747809485		Iterasi 4	0.747809485
	Iterasi 5	0.197179976		Iterasi 5	0.464786651
	Iterasi 6	-0.412007791		Iterasi 6	-0.129606232
AND	Iterasi 1	0.769054388	XOR	Iterasi 1	0.538108777519 1024
	Iterasi 2	0.759194000			
	Iterasi 4	0.747809485			
	Iterasi 5	0.732393325			
	Iterasi 6	0.152795325			
	Iterasi 1	-0.779953141			
SEQUENCE					
Percobaan 1	Model B / Model A	Kepegawaian Mahasiwa	Percobaan 2	Model B / Model A	Kepegawaian Mahasiswa
	Kepegawaian *Salaf*	0.779933590		Kepegawaian *Salaf*	0.779933590
	Iterasi 1	0.762378607		Iterasi 1	0.742595821
	Iterasi 2	0.485191643		Iterasi 2	0.717598441
	Iterasi 3	0.152795325		Iterasi 3	0.694100529
	Iterasi 4			Iterasi 4	0.671800842
	Iterasi 5			Iterasi 5	0.396857130
	Iterasi 6			Iterasi 6	-0.336944833
	Iterasi 7			Iterasi 7	

Menurut **Tabel 4.30** di atas, simulasi pertumbuhan percobaan pertama dan kedua pada pembobotan percabangan dan *sequence* mendapatkan hasil iterasi yang berbeda. Seperti yang telah dipaparkan, nilai *scalability* awal pada perbandingan proses bisnis Kepegawaian *Salaf* pada Gambar 3.6 dan proses bisnis Kepegawaian Mahasiswa pada Gambar 3.16 adalah 0.779933590, maka simulasi pertumbuhan dapat terjadi.

Berikut ditunjukkan hasil simulasi pertumbuhan dari Kepegawaian *Salaf* sebagai *input*-an A dan Kepegawaian *Tahfidz* sebagai *input*-an B. Perbandingan antar model tersebut, *input*-an A mewakili proses bisnis

sequence, sedangkan *input*-an B mewakili proses bisnis bercabang (level3). Hasil perbandingan pada **Tabel 4.31** di bawah ini: X

Tabel 4.31 Simulasi Pertumbuhan Kepegawaian *Salaf* & Kepegawaian *Tahfidz*

PERCABANGAN					
Percobaan 1	**Model B / Model A**	**Kepegawaian *Tahfidz***	Percobaan 2	**Model B / Model A**	**Kepegawaian *Tahfidz***
	Kepegawaian *Salaf*	0.973988914		**Kepegawaian *Salaf***	0.973988914
	Iterasi 1	0.972650944		Iterasi 1	0.972650944
	Iterasi 2	0.971416848		Iterasi 2	0.969206889
	Iterasi 3	0.970272623		Iterasi 3	0.872841011
	Iterasi 4	0.969206889		Iterasi 4	0.869099445
	Iterasi 5	0.872841011		Iterasi 5	0.800470263
	Iterasi 6	0.869099445		Iterasi 6	0.969206889
	Iterasi 7	0.800470263		Iterasi 7	0.872841011
	Iterasi 8	0.791764111		Iterasi 8	
	Iterasi 9			Iterasi 9	
SEQUENCE					
Percobaan 1	**Model B / Model A**	**Kepegawaian *Tahfidz***	Percobaan 2	**Model B / Model A**	**Kepegawaian *Tahfidz***
	Kepegawaian *Salaf*	0.973988914		**Kepegawaian *Salaf***	0.973988914
	Iterasi 1	0.971817598		Iterasi 1	0.971817598
	Iterasi 2	0.969909156		Iterasi 2	0.969909156
	Iterasi 3	0.904630758		Iterasi 3	0.936420505
	Iterasi 4	0.866727889		Iterasi 4	0.900045917
	Iterasi 5	0.790473646		Iterasi 5	0.790473646
	Iterasi 6	0.779149004		Iterasi 6	
	Iterasi 7			Iterasi 7	

Menurut **Tabel 4.31** di atas, simulasi pertumbuhan percobaan pertama dan kedua pada pembobotan percabangan dan *sequence* mendapatkan hasil iterasi yang berbeda. Seperti yang telah dipaparkan, nilai *scalability* awal pada perbandingan proses bisnis Kepegawaian *Salaf* pada Gambar 3.6 dan proses bisnis Kepegawaian *Tahfidz* pada Gambar 3.21 adalah 0.973988914, maka simulasi pertumbuhan dapat terjadi.

Berikut ditunjukkan hasil simulasi pertumbuhan dari Kepegawaian *Salaf* sebagai *input*-an A dan Kepegawaian *Modern* sebagai *input*-an B. Perbandingan antar model tersebut, *input*-an A mewakili proses bisnis

bercabang (level3), sedangkan *input*-an B mewakili proses bisnis bercabang (level3). Hasil perbandingan pada **Tabel 4.32** di bawah ini: X

Tabel 4.32 Simulasi Pertumbuhan Kepegawaian *Salaf* & Kepegawaian *Modern*

PERCABANGAN					
Percobaan 1	**Model B / Model A**	**Kepegawaian *Modern***	Percobaan 2	**Model B / Model A**	**Kepegawaian *Modern***
	Kepegawaian *Salaf*	0.973939673		**Kepegawaian *Salaf***	0.973939673
	Iterasi 1	0.972605444		Iterasi 1	0.971366943
	Iterasi 2	0.971366943		Iterasi 2	0.910636196
	Iterasi 3	0.910636196		Iterasi 3	0.876523601
	Iterasi 4	0.876523601		Iterasi 4	0.872460790
	Iterasi 5	0.872460790		Iterasi 5	0.770106001
	Iterasi 6	0.770106001		Iterasi 6	0.672721837
	Iterasi 7	0.699019686		Iterasi 7	0.658333333
	Iterasi 8	0.672721837		Iterasi 8	0.629606506
	Iterasi 9	0.658333333		Iterasi 9	0.612830075
	Iterasi 10	0.629606506		Iterasi 10	
	Iterasi 11	0.612830075		Iterasi 11	
	Iterasi 12	0.595804145		Iterasi 12	
	Iterasi 13			Iterasi 13	
SEQUENCE					
Percobaan 1	**Model B / Model A**	**Kepegawaian *Modern***	Percobaan 2	**Model B / Model A**	**Kepegawaian *Modern***
	Kepegawaian *Salaf*	0.973939673		**Kepegawaian *Salaf***	0.973939673
	Iterasi 1	0.971769939		Iterasi 1	0.971769939
	Iterasi 2	0.969843924		Iterasi 2	
	Iterasi 3				
	Iterasi 5				
	Iterasi 6				
	Iterasi 7				

Menurut **Tabel 4.32** di atas, simulasi pertumbuhan percobaan pertama dan kedua pada pembobotan percabangan dan *sequence* mendapatkan hasil iterasi yang berbeda. Seperti yang telah dipaparkan, nilai *scalability* awal pada perbandingan proses bisnis Kepegawaian *Salaf* pada Gambar 3.6 dan proses bisnis Kepegawaian *Modern* pada Gambar 3.11 adalah 0.973939673, maka simulasi pertumbuhan dapat terjadi.

Berikut ditunjukkan hasil simulasi pertumbuhan dari Kepegawaian Mahasiswa sebagai *input*-an A dan Kepegawaian *Tahfidz* sebagai *input*-an B.

Perbandingan antar model tersebut, *input*-an A mewakili proses bisnis bercabang (level2), sedangkan *input*-an B mewakili proses bisnis bercabang (level3). Hasil perbandingan pada **Tabel 4.33** di bawah ini:

Tabel 4.33 Simulasi Pertumbuhan Kepegawaian Mahasiswa & Kepegawaian *Tahfidz*

PERCABANGAN					
	Model A \ Model B	Kepegawaian *Tahfidz*		Model A \ Model B	Kepegawaian *Tahfidz*
Percobaan 1	**Kepegawaian Mahasiswa**	0.889639114	Percobaan 2	**Kesantrian Mahasiswa**	0.889639114
	Iterasi 1	0.885769878		Iterasi 1	0.882711442
	Iterasi 2	0.764355504		Iterasi 2	0.757654746
	Iterasi 3	0.757654746		Iterasi 3	0.751379102
	Iterasi 4	0.751379102		Iterasi 4	0.530845771
	Iterasi 5	0.637332705		Iterasi 5	0.439099720
	Iterasi 6	0.530845771		Iterasi 6	0.069723629
	Iterasi 7	0.214739608		Iterasi 7	-0.082541167
	Iterasi 8	-0.046560916		Iterasi 8	
SEQUENCE					
	Model A \ Model B	Kepegawaian *Tahfidz*		Model A \ Model B	Kepegawaian *Tahfidz*
Percobaan 1	**Kepegawaian Mahasiswa**	0.889639114	Percobaan 2	**Kesantrian Mahasiswa**	0.889639114
	Iterasi 1	0.883346662		Iterasi 1	0.883346662
	Iterasi 2	0.755518214		Iterasi 2	0.877759107
	Iterasi 3	0.618219738		Iterasi 3	0.618219738
	Iterasi 4	0.466999632		Iterasi 4	0.600249724
	Iterasi 5	0.114713601		Iterasi 5	0.494122057
	Iterasi 6			Iterasi 6	0.286234167
	Iterasi 7			Iterasi 7	0.003325649

Menurut **Tabel 4.33** di atas, simulasi pertumbuhan percobaan pertama dan kedua pada pembobotan percabangan dan *sequence* mendapatkan hasil iterasi yang berbeda. Seperti yang telah dipaparkan, nilai *scalability* awal pada perbandingan proses bisnis Kepegawaian Mahasiswa pada Gambar 3.16 dan proses bisnis Kepegawaian *Tahfidz* pada Gambar 3.21 adalah 0.889639114, maka simulasi pertumbuhan dapat terjadi.

Berikut ditunjukkan hasil simulasi pertumbuhan dari Kepegawaian Mahasiswa sebagai *input*-an A dan Kepegawaian *Modern* sebagai *input*-an B pada **Tabel 4.34** di bawah ini:

Tabel 4.34 Simulasi Pertumbuhan Kepegawaian Mahasiswa & Kepegawaian *Modern*

PERCABANGAN					
Percobaan 1	Model B / Model A	Kepegawaian *Modern*	Percobaan 2	Model B / Model A	Kepegawaian *Modern*
	Kepegawaian Mahasiswa	0.936562248		**Kepegawaian Mahasiswa**	0.936562248
	Iterasi 1	0.934212536		Iterasi 1	0.934212536
	Iterasi 2	0.932013572		Iterasi 2	0.932013572
	Iterasi 3	0.859896753		Iterasi 3	0.719793506
	Iterasi 4	0.712010453		Iterasi 4	0.712010453
	Iterasi 5	0.723061992		Iterasi 5	0.653827490
	Iterasi 6	0.718540217		Iterasi 6	0.507445380
	Iterasi 7	0.595877735		Iterasi 7	0.352109859
	Iterasi 8	0.449309940		Iterasi 8	0.191165237
	Iterasi 9	0.297522110		Iterasi 9	0.100000009
	Iterasi 10	0.132121773		Iterasi 10	0.002719987
SEQUENCE					
Percobaan 1	Model B / Model A	Kepegawaian *Modern*	Percobaan 2	Model B / Model A	Kepegawaian *Modern*
	Kepegawaian Mahasiswa	0.936562248		**Kepegawaian Mahasiswa**	0.936562248
	Iterasi 1	0.932730919		Iterasi 1	0.932730919
	Iterasi 2	0.929287161		Iterasi 2	0.929287161
	Iterasi 3	0.778492185		Iterasi 3	0.704656246
	Iterasi 4	0.543168912		Iterasi 4	
	Iterasi 5	0.411756536		Iterasi 5	
	Iterasi 6			Iterasi 6	

Menurut **Tabel 4.34** di atas, simulasi pertumbuhan percobaan pertama dan kedua pada pembobotan percabangan dan *sequence* mendapatkan hasil iterasi yang berbeda. Seperti yang telah dipaparkan, nilai *scalability* awal pada perbandingan proses bisnis Kepegawaian Mahasiswa pada Gambar 3.16 dan proses bisnis Kepegawaian *Modern* pada Gambar 3.11 adalah 0.936562248, maka simulasi pertumbuhan dapat terjadi.

Berikut ditunjukkan hasil simulasi pertumbuhan dari Kepegawaian *Tahfidz* sebagai *input*-an A dan Kepegawaian *Modern* sebagai *input*-an B pada **Tabel 4.35** di bawah ini:

Tabel 4.35 Simulasi Pertumbuhan Kepegawaian *Tahfidz* & Kepegawaian *Modern*

PERCABANGAN					
Percobaan 1	Model B / Model A	Kepegawaian *Modern*	Percobaan 2	Model B / Model A	Kepegawaian *Modern*
	Kepegawaian *Tahfidz*	0.586118495		Kepegawaian *Tahfidz*	0.586118495
	Iterasi 1	0.471037754		Iterasi 1	0.559037234
	Iterasi 2	0.382517308		Iterasi 2	0.491196809
	Iterasi 3	0.300663992		Iterasi 3	0.423356383
	Iterasi 4			Iterasi 4	0.389436170
	Iterasi 5			Iterasi 5	0.355515958
	Iterasi 6			Iterasi 6	0.287675532
	Iterasi 7			Iterasi 7	0.151994681
	Iterasi 8			Iterasi 8	0.081106824
	Iterasi 9			Iterasi 9	
AND	Iterasi 1	0.576830203	XOR		-
	Iterasi 2	0.541398944			-
	Iterasi 3	0.503182189			-
	Iterasi 4	0.491196809			-
	Iterasi 5	0.457276596			-
	Iterasi 6	0.389436170			-
	Iterasi 7	0.321595745			-
	Iterasi 8	0.287675532			-
	Iterasi 9	0.253755319			-
	Iterasi 10	0.219835107			-
	Iterasi 11	0.151994681			-
	Iterasi 12	0.118074469			-
	Iterasi 13	0.050234043			-
	Iterasi 14	-0.017606381			-
SEQUENCE					
Percobaan 1	Model B / Model A	Kepegawaian *Modern*	Percobaan 2	Model B / Model A	Kepegawaian *Modern*
	Kepegawaian *Tahfidz*	0.586118495		Kepegawaian *Tahfidz*	0.586118495
	Iterasi 1	0.467301032		Iterasi 1	0.478528045
	Iterasi 2	0.374233655		Iterasi 2	0.467301032
	Iterasi 3	0.090165703		Iterasi 3	
	Iterasi 4			Iterasi 4	

Menurut **Tabel 4.35** di atas, simulasi pertumbuhan percobaan pertama dan kedua pada pembobotan percabangan dan *sequence* mendapatkan hasil iterasi yang berbeda. Seperti yang telah dipaparkan, nilai *scalability* awal pada perbandingan proses bisnis Kepegawaian *Tahfidz* pada Gambar 3.21 dan proses bisnis Kepegawaian *Modern* pada Gambar 3.11 adalah 0.586118495, maka simulasi pertumbuhan dapat terjadi.

4.7 Pengukuran Hasil Pengujian

4.7.1 Nilai *Scalability*

Pengukuran hasil pengujian yaitu mengukur akurasi penelitian dengan acuan nilai minimum *scalability* hasil dari simulasi pertumbuhan model PNML dari beberapa percobaan yang telah dilakukan. Cara mengukur akurasi nilai minimum yang dirumuskan seperti pada persamaan 4.1 di bawah ini:

$$s = \sqrt{\frac{n\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2}{n\,(n-1)}} \quad \ldots\ldots\ldots (4.1)$$

Keterangan:
s^2 : Varian
s : Standar deviasi
x_i : Nilai x ke-i
$\bar{x}$: Rata-rata
n : Ukuran sampel

Pengukuran standart deviasi dilakukan pada masing-masing bagian Pondok Pesantren, di antaranya:

1. Bagian Akademik

a. Akademik Percabangan

Akumulasi dari nilai *scalability* minimum pada bagian Akademik dengan pembobotan percabangan ditunjukkan pada **Tabel 4.36** sebagai berikut:

Tabel 4.36 Akumulasi Nilai *similarity* Akademik Percabangan

No	x_i	x_i^2
1	0.118032610	0.013931697
2	0.096852116	0.009380332
3	0.083784589	0.007019857
4	0.083784581	0.007019856
5	0.208961765	0.043665019
6	0.208961765	0.043665019
7	0.291291366	0.084850660
8	0.055055155	0.003031070
9	0.157513295	0.024810438
10	0.126977880	0.016123382
11	0.208399391	0.043430306

No	x_i	x_i^2
12	0.178263965	0.031778041
13	0.067503302	0.004556696
14	0.067503302	0.004556696
Σ	1.952885082	0.337819070

$(\sum_{i=1}^{n} x_i)^2 = 1.952885082^2 = 3.813760145$

$$s^2 = \frac{(14).(0.33781907) - (3.813760145)}{(14).(13)}$$

$$= \frac{(4.729466984) - (3.813760145)}{(14).(13)}$$

$s^2 = 0.91570684 / 182 = 0.00503135$

$s = 0.07093$

Berdasarkan hasil dari perhitungan nilai *scalability* minimum setelah simulasi pertumbuhan pada bagian Akademik dengan pembobotan percabangan menggunakan *standart* deviasi, menunjukkan bahwa didapatkan nilai *standart* deviasi adalah 0.07093.

b. Akademik *Sequence*

Akumulasi dari nilai *scalability* minimum pada bagian Akademik dengan pembobotan *sequence* ditunjukkan pada **Tabel 4.37** sebagai berikut:

Tabel 4.37 Akumulasi Nilai *Scalability* Akademik *Sequence*

No	x_i	x_i^2
1	0.092134523	0.008488770
2	0.092134523	0.008488770
3	0.348102792	0.121175554
4	0.091510681	0.008374205
5	0.208961765	0.043665019
6	0.208961765	0.043665019
7	0.136737759	0.018697215
8	0.019838484	0.000393565
9	0.125767119	0.015817368
10	0.126977880	0.016123382
11	0.189969094	0.036088257
12	0.189969094	0.036088257
13	0.118871764	0.014130496
14	0.118871764	0.014130496
Σ	2.068809007	0.385326374

$(\sum_{i=1}^{n} x_i)^2$ = 2.068809007 ^{2} = 4.279970707

$$s^2 = \frac{(14).(0.38532634) - (4.279970707)}{(14).(13)}$$

$$= \frac{(5.394569236) - (4.279970707)}{(14).(13)}$$

s^2 = 1.114598528/182 = 0.00612416

s = 0.07825

Berdasarkan hasil dari perhitungan nilai *scalability* minimum setelah simulasi pertumbuhan pada bagian Akademik dengan pembobotan *sequence* menggunakan *standart* deviasi, menunjukkan bahwa didapatkan nilai *standart* deviasi adalah 0.07825.

2. Bagian PSB

a. PSB Percabangan

Akumulasi dari nilai *scalability* minimum pada bagian PSB dengan pembobotan percabangan ditunjukkan pada **Tabel 4.38** di bawah ini:

Tabel 4.38 Akumulasi Nilai *Scalability* PSB Percabangan

No	x_i	x_i^2
1	0.107142857	0.011479592
2	0.039639952	0.001571326
3	0.298060054	0.088839796
4	0.199244047	0.03969819
5	0.104136463	0.010844403
6	0.325136099	0.105713483
7	0.055315809	0.003059839
8	0.161294172	0.026015810
9	0.438975908	0.192699848
10	0.479160586	0.229594867
11	0.291710389	0.085094951
12	0.291710389	0.085094951
Σ	2.791526725	0.879707055

$(\sum_{i=1}^{n} x_i)^2$ = 2.791526725 ^{2} = 7.792621456

$$s^2 = \frac{(12).(0.879707055) - (7.792621456)}{(12).(11)}$$

$$= \frac{(10.55648466) - (7.792621456)}{(12).(11)}$$

s^2 = 2.763863206/132 = 0.020938358

s = 0.14470

Berdasarkan hasil dari perhitungan nilai *scalability* minimum setelah simulasi pertumbuhan pada bagian PSB dengan pembobotan percabangan menggunakan *standart* deviasi, menunjukkan bahwa didapatkan nilai *standart* deviasi adalah 0.14470.

b. PSB *Sequence*

Akumulasi dari nilai *scalability* minimum pada bagian PSB dengan pembobotan *sequence* ditunjukkan pada **Tabel 4.39** di bawah ini:

Tabel 4.39 Akumulasi Nilai *Scalability* PSB *Sequence*

No	x_i	x_i^2
1	0.474363040	0.225020294
2	0.211544560	0.044751101
3	0.181486344	0.032937293
4	0.117794128	0.013875457
5	0.216432263	0.046842924
6	0.269376531	0.072563715
7	0.290332232	0.084292805
8	0.064702429	0.004186404
9	0.362354051	0.131300458
10	0.454800429	0.206843430
11	0.217984614	0.047517292
12	0.217984614	0.047517292
Σ	3.079155235	0.957648466

$(\sum_{i=1}^{n} x_i)^2$ = 3.079155235^2 = 9.481196861

$$s^2 = \frac{(12).(0.957648466) - (9.481196861)}{(12).(11)}$$

$$= \frac{(11.49178159) - (9.481196861)}{(12).(11)}$$

s^2 = 2.010584628 /132 = 0.015231702

s = 0.12341

Berdasarkan hasil dari perhitungan nilai *scalability* minimum setelah simulasi pertumbuhan pada bagian Akademik dengan pembobotan *sequence* menggunakan *standart* deviasi, menunjukkan bahwa didapatkan nilai *standart* deviasi adalah 0.12341.

3. Bagian Kesantrian

a. Kesantrian Percabangan

Akumulasi dari nilai *scalability* minimum pada bagian Kesantrian dengan pembobotan percabangan ditunjukkan pada **Tabel 4.40** di bawah ini:

Tabel 4.40 Akumulasi Nilai *Scalability* Kesantrian Percabangan

No	x_i	x_i^2
1	0.056820481	0.003228567
2	0.067008307	0.004490113
3	0.043812340	0.001919521
4	0.126176898	0.015920610
5	0.004887985	2.38924E-05
6	0.260929584	0.068084248
7	0.192134585	0.036915699
8	0.004609657	2.12489E-05
9	0.063324737	0.004010022
10	0.003810828	1.45224E-05
11	0.245337601	0.060190538
12	0.001692486	2.86451E-06
Σ	1.070545489	0.194821847

$(\sum_{i=1}^{n} x_i)^2 = 1.070545489^{\,2} = 1.146067644$

$$s^2 = \frac{(12).(1.210721380) - (1.146067644)}{(12).(11)}$$

$$= \frac{(2.337862159) - (1.146067644)}{(12).(11)}$$

s^2 = 1.19194515/132 = 0.009028746

s = 0.09501

Berdasarkan hasil dari perhitungan nilai *scalability* minimum setelah simulasi pertumbuhan pada bagian Kesantrian dengan pembobotan

percabangan menggunakan *standart* deviasi, menunjukkan bahwa didapatkan nilai *standart* deviasi adalah 0.09501.

b. Kesantrian *Sequence*

Akumulasi dari nilai *scalability* minimum pada bagian Kesantrian dengan pembobotan *sequence* ditunjukkan pada **Tabel 4.41** di bawah ini:

Tabel 4.41 Akumulasi Nilai *Scalability* Kesantrian *Sequence*

No	x_i	x_i^2
1	0.363071680	0.131821045
2	0.132089580	0.017447657
3	0.603120919	0.363754843
4	0.603120919	0.363754843
5	0.108865473	0.011851691
6	0.152192670	0.023162609
7	0.583123503	0.340033020
8	0.281690375	0.079349467
9	0.154230861	0.023787158
10	0.029841316	0.000890504
11	0.022859146	0.000522541
12	0.355239431	0.126195053
Σ	3.389445873	1.482570431

$$(\sum_{i=1}^{n} x_i)^2 = 3.389445873^{\,2} = 11.48834333$$

$$s^2 = \frac{(12).(1.482570431) - (11.48834333)}{(12).(11)}$$

$$= \frac{(17.79084518) - (11.48834333)}{(12).(11)}$$

s^2 = 6.302501852 /132 = 0.047746226

s = 0.21850

Berdasarkan hasil dari perhitungan nilai *scalability* minimum setelah simulasi pertumbuhan pada bagian Kesantrian dengan pembobotan *sequence* menggunakan *standart* deviasi, menunjukkan bahwa didapatkan nilai *standart* deviasi adalah 0.21850.

4. Bagian Sarpras

a. Sarpras Percabangan

Akumulasi dari nilai *scalability* minimum pada bagian Sarpras dengan pembobotan percabangan ditunjukkan pada **Tabel 4.42** sebagai berikut:

Tabel 4.42 Akumulasi Nilai *Scalability* Sarpras Percabangan

No	x_i	x_i^2
1	0.421982758	0.178069448
2	0.322690051	0.104128869
3	0.389407312	0.151638055
4	0.443452669	0.196650270
5	0.291992727	0.085259753
6	0.354466534	0.125646524
7	0.443452669	0.196650270
8	0.371933090	0.138334223
9	0.257502577	0.066307577
10	0.628751288	0.395328182
11	0.348383144	0.121370815
12	0.203573048	0.041441986
13	0.163768737	0.026820199
14	0.285927147	0.081754333
Σ	4.927283751	1.909400504

$(\sum_{i=1}^{n} x_i)^2 = 4.97283751^{\,2} = 24.27812516$

$$s^2 = \frac{(14).(1.909400504) - (24.27812516)}{(14).(13)}$$

$$= \frac{(26.73160705) - (24.27812516)}{(14).(13)}$$

$s^2 = 2.453481888 / 182 = 0.01348067$

$s = 0.11610$

Berdasarkan hasil dari perhitungan nilai *scalability* minimum setelah simulasi pertumbuhan pada bagian Sarpras dengan pembobotan percabangan menggunakan *standart* deviasi, menunjukkan bahwa didapatkan nilai *standart* deviasi adalah 0.11610.

b. Sarpras *Sequence*

Akumulasi dari nilai *scalability* minimum pada bagian Sarpras dengan pembobotan *sequence* ditunjukkan pada **Tabel 4.43** di bawah ini:

Tabel 4.43 Akumulasi Nilai *Scalability* Sarpras *Sequence*

No	x_i	x_i^2
1	0.312345158	0.097559498
2	0.365326980	0.133463802
3	0.407133619	0.165757784
4	0.452736736	0.204970552
5	0.312345158	0.097559498
6	0.365326980	0.133463802
7	0.407133619	0.165757784
8	0.363283212	0.131974692
9	0.192783570	0.037165505
10	0.596391785	0.355683161
11	0.257263975	0.066184753
12	0.050921550	0.002593004
13	0.202951407	0.041189274
14	0.017906165	0.000320631
Σ	4.303849914	1.633643739

$(\sum_{i=1}^{n} x_i)^2 = 4.303849914^2 = 18.52312408$

$$s^2 = \frac{(14).(1.633643739) - (18.52312408)}{(14).(13)}$$

$$= \frac{(22.87101235) - (18.52312408)}{(14).(13)}$$

$s^2 = 4.347888268 / 182 = 0.023889496$

$s = 0.15456$

Berdasarkan hasil dari perhitungan nilai *scalability* minimum setelah simulasi pertumbuhan pada bagian Sarpras dengan pembobotan *sequence* menggunakan *standart* deviasi, menunjukkan bahwa didapatkan nilai *standart* deviasi adalah 0.15456.

5. Bagian Kepegawaian

a. Kepegawaian Percabangan

Akumulasi dari nilai *scalability* minimum pada bagian Kepegawaian dengan pembobotan percabangan ditunjukkan pada **Tabel 4.44** di bawah ini:

Tabel 4.44 Akumulasi Nilai *Scalability* Kepegawaian Percabangan

No	x_i	x_i^2
1	0.132121773	0.017456163
2	0.002719987	7.39833E-06

No	x_i	x_i^2
3	0.214739608	0.046113099
4	0.069723629	0.004861384
5	0.197179976	0.038879943
6	0.243428456	0.059257413
7	0.595804145	0.354982579
8	0.612830075	0.375560701
9	0.791764111	0.626890407
10	0.800470263	0.640752642
11	0.300663992	0.090398836
12	0.081106824	0.006578317
Σ	4.042552839	2.261738883

$(\sum_{i=1}^{n} x_i)^2 = 4.042552839^{\,2} = 16.34223346$

$$s^2 = \frac{(12).(2.261738883) - (16.34223346)}{(12).(11)}$$

$$= \frac{(27.1408666) - (16.34223346)}{(12).(11)}$$

s^2 = 10.79863315/132 = 0.081807827

s = 0.28602

Berdasarkan hasil dari perhitungan nilai *scalability* minimum setelah simulasi pertumbuhan pada bagian Kepegawaian dengan pembobotan percabangan menggunakan *standart* deviasi, menunjukkan bahwa didapatkan nilai *standart* deviasi adalah 0.28602.

b. Kepegawaian *Sequence*

Akumulasi dari nilai *scalability* minimum pada bagian Kepegawaian dengan pembobotan *sequence* ditunjukkan pada **Tabel 4.45** di bawah ini:

Tabel 4.45 Akumulasi Nilai *Scalability* Kepegawaian *Sequence*

No	x_i	x_i^2
1	0.411756536	0.169543445
2	0.704656246	0.496540425
3	0.114713601	0.013159210
4	0.003325649	1.10599E-05
5	0.152795325	0.023346411
6	0.396857130	0.157495582
7	0.969843924	0.940597237
8	0.971769939	0.944336814
9	0.779149004	0.607073170

No	x_i	x_i^2
10	0.790473646	0.624848585
11	0.090165703	0.008129854
12	0.467301032	0.218370255
Σ	5.852807735	4.203452048

$(\sum_{i=1}^{n} x_i)^2 = 5.852807735^2 = 34.25535838$

$$s^2 = \frac{(12).(4.203452048) - (34.25535838)}{(12).(11)}$$

$$= \frac{(50.44142458) - (34.25535838)}{(12).(11)}$$

$s^2 = 16.1860662/132 = 0.122621714$

$s = 0.35017$

Berdasarkan hasil dari perhitungan nilai *scalability* minimum setelah simulasi pertumbuhan pada bagian Kepegawaian dengan pembobotan *sequence* menggunakan *standart* deviasi, menunjukkan bahwa didapatkan nilai *standart* deviasi adalah 0.35017.

Berdasarkan hasil dari perhitungan standart deviasi dari berbagai bagian Pondok Pesantren, dapat dikatakan penelitian cukup berhasil karena rata-rata yang dihasilkan pada perhitungan di atas bernilai "<0.2".

4.7.2 Nilai *Similarity*

Pengukuran hasil pengujian dengan nilai *similarity* adalah mengukur akurasi penelitian dengan acuan nilai *similarity* hasil dari simulasi pertumbuhan model PNML dari beberapa percobaan yang telah dilakukan. Pengukuran dilakukan dengan *input*-an A adalah hasil dari simulasi pertumbuhan, sedangkan *input*-an B adalah model B yang digunakan sebagai pembanding. Akurasi dapat dilihat dari rata-rata nilai *similarity* yang dihasilkan yang dirumuskan dengan persamaan 4.2 di bawah ini:

$$\overline{sim} = \frac{\sum sim}{n}$$

Keterangan:
sim = Nilai *similarity*

n = Ukuran sampel

$\overline{sim}$ = Rata-rata nilai *similarity*

1. Proses Bisnis *Sequence* – Proses Bisnis *Sequence*

Pengukuran nilai *similarity* dua model PNML yakni PSB *Salaf* sebagai *input*-an A dan PSB Mahasiswa sebagai *input*-an B. Perbandingan menunjukkan *input*-an A merupakan proses bisnis *sequence*, sedangkan *input*-an B juga proses bisnis *sequence*. Ditunjukkan pada **Tabel 4.46** di bawah ini:

Tabel 4.46 Akumulasi Nilai *Similarity* Proses Bisnis *Sequence* - *Sequence*

n	sim
1	0.080681818
2	0.097140512
3	0.193072224
4	0.123200749
5	0.066566375
Σ	0.560661678

$$\overline{sim} = \frac{0.560661678}{5}$$
$$= 0.112132336$$

Berdasarkan **Tabel 4.46** di atas, nilai *sim* didapatkan dari input-an A (hasil dari simulasi pertumbuhan PSB *Salaf* dan PSB Mahasiswa) dan *input*-an B (PSB Mahasiswa). Rata-rata *similarity* yang didapatkan adalah 0.112132336

2. Proses Bisnis *Sequence* – Proses Bisnis Bercabang (level1)

Pengukuran nilai *similarity* dua model PNML yakni Kepegawaian *Salaf* sebagai *input*-an A dan Kepegawaian Mahasiswa sebagai *input*-an B. Perbandingan menunjukkan *input*-an A merupakan proses bisnis *sequence*,

sedangkan *input*-an B juga proses bisnis bercabang (level1). Ditunjukkan pada **Tabel 4.47** di bawah ini:

Tabel 4.47 Akumulasi Nilai *Similarity* Proses Bisnis *Sequence* – Bercabang (level1)

n	sim
1	0.131945194
2	0.223821102
3	0.168424169
4	0.227269933
5	0.163221778
Σ	0.914682176

$$\overline{sim} = \frac{0.914682176}{5}$$
$$= 0.182936435$$

Berdasarkan **Tabel 4.47** di atas, nilai *sim* didapatkan dari input-an A (hasil dari simulasi pertumbuhan Kepegawaian *Salaf* dan Kepegawaian Mahasiswa) dan *input*-an B (Kepegawaian Mahasiswa). Rata-rata *similarity* yang didapatkan adalah 0.182936435

3. Proses Bisnis *Sequence* – Proses Bisnis Bercabang (level2)

Pengukuran nilai *similarity* dua model PNML yakni Kesantrian *Salaf* sebagai *input*-an A dan Kesantrian *Tahfidz* sebagai *input*-an B. Perbandingan menunjukkan *input*-an A merupakan proses bisnis *sequence*, sedangkan *input*-an B juga proses bisnis bercabang (level2). Ditunjukkan pada **Tabel 4.48** di bawah ini:

Tabel 4.48 Akumulasi Nilai *Similarity* Proses Bisnis *Sequence* – Bercabang (level2)

n	sim
1	0.223086705
2	0.098654700
3	0.294518549
4	0.209015275
5	0.199063260
Σ	1.024338489

$$\overline{sim} = \frac{1.024338489}{5}$$
$$= 0.204867698$$

Berdasarkan **Tabel 4.48** di atas, nilai *sim* didapatkan dari input-an A (hasil dari simulasi pertumbuhan Kesantrian *Salaf* dan Kesantrian *Tahfidz*) dan *input*-an B (Kesantrian *Tahfidz*). Rata-rata *similarity* yang didapatkan adalah 0.204867698

4. Proses Bisnis Bercabang (level1) – Proses Bisnis Bercabang (level1)

Pengukuran nilai *similarity* dua model PNML yakni Akademik *Tahfidz* sebagai *input*-an A dan Akademik Mahasiswa sebagai *input*-an B. Perbandingan menunjukkan *input*-an A merupakan proses bisnis bercabang (level1), sedangkan *input*-an B juga proses bisnis bercabang (level1). Ditunjukkan pada **Tabel 4.49** di bawah ini:

Tabel 4.49 Akumulasi Nilai *Similarity* Proses Bisnis Bercabang (level1) – Bercabang (level1)

n	sim
1	0.242074039
2	0.233041021
3	0.247305284
4	0.228372988
5	0.406648078
Σ	1.357441410

$$\overline{sim} = \frac{1.357441410}{5}$$
$$= 0.271488282$$

Berdasarkan **Tabel 4.49** di atas, nilai *sim* didapatkan dari input-an A (hasil dari simulasi pertumbuhan Akademik *Tahfidz* dan Akademik Mahasiswa) dan *input*-an B (Akademik Mahasiswa). Rata-rata *similarity* yang didapatkan adalah 0.271488282

5. Proses Bisnis Bercabang (level2) – Proses Bisnis Bercabang (level2)

Pengukuran nilai *similarity* dua model PNML yakni Kesantrian Mahasiswa sebagai *input*-an A dan Kesantrian *Tahfidz* sebagai *input*-an B. Perbandingan menunjukkan *input*-an A merupakan proses bisnis bercabang (level2), sedangkan *input*-an B juga proses bisnis bercabang (level2). Ditunjukkan pada **Tabel 4.50** di bawah ini:

Tabel 4.50 Akumulasi Nilai *Similarity* Proses Bisnis Bercabang (level2) – Bercabang (level2)

n	sim
1	0.190939109
2	0.202598193
3	0.229646712
4	0.163848193
5	0.108273403
Σ	0.895305610

$$\overline{sim} = \frac{0.89530561}{5}$$
$$= 0.179061122$$

Berdasarkan **Tabel 4.50** di atas, nilai *sim* didapatkan dari input-an A (hasil dari simulasi pertumbuhan Kesantrian Mahasiswa dan Kesantrian *Tahfidz*) dan *input*-an B (Kesantrian *Tahfidz*). Rata-rata *similarity* yang didapatkan adalah 0.179061122.

Beberapa pengukuran pengujian yang telah dilakukan dari segi nilai *similarity*, hasil simulasi pertumbuhan menunjukkan rata-rata nilai *similarity* "<0.5". Berdasarkan hasil rata-rata *similarity* yang didapatkan menunjukkan bahwa penelitian cenderung belum maksimal jika pertumbuhan proses bisnis bersifat *random*.

4.8 Integrasi Penelitian dengan Islam

Penelitan yang dihasilkan marupakan model proses bisnis yang telah disimulasikan pertumbuhannya. Simulasi pertumbuhan proses bisnis maksudnya adalah menumbuhakan elemen baru dalam proses bisnis. Model proses dalam penelitian ini adalah sebuah *graph* yang mana nantinya diselesaikan dengan aturan produksi *Cellular Automata*. Aturan tersebeut berkaitan dengan langkah untuk menumbuhkan elemen baru dalam proses bisnis. Pertumbuhan proses bisnis sendiri dianalogikan seperti pertumbuhan tanaman, yaitu dengan tumbuhnya cabang baru dalam model proses bisnis. Allah sendiri telah berfirman dalam Q.S Al-an'am ayat 99 yang berbunyi:

وَهُوَ الَّذِي أَنْزَلَ مِنَ السَّمَاءِ مَاءً فَأَخْرَجْنَا بِهِ نَبَاتَ كُلِّ شَيْءٍ فَأَخْرَجْنَا مِنْهُ خَضِرًا نُخْرِجُ مِنْهُ حَبًّا مُتَرَاكِبًا وَمِنَ النَّخْلِ مِنْ طَلْعِهَا قِنْوَانٌ دَانِيَةٌ وَجَنَّاتٍ مِنْ أَعْنَابٍ وَالزَّيْتُونَ وَالرُّمَّانَ مُشْتَبِهًا وَغَيْرَ مُتَشَابِهٍ ۗ انْظُرُوا إِلَىٰ ثَمَرِهِ إِذَا أَثْمَرَ وَيَنْعِهِ ۚ إِنَّ فِي ذَٰلِكُمْ لَآيَاتٍ لِقَوْمٍ يُؤْمِنُونَ

Artinya:

Dan Dialah yang menurunkan air hujan dari langit, lalu Kami tumbuhkan dengan air itu segala macam tumbuh-tumbuhan maka Kami keluarkan dari tumbuh-tumbuhan itu tanaman yang menghijau. Kami keluarkan dari tanaman yang menghijau itu butir yang banyak; dan dari mayang korma mengurai tangkai-tangkai yang menjulai, dan kebun-kebun anggur, dan (Kami keluarkan pula) zaitun dan delima yang serupa dan yang tidak serupa. Perhatikanlah buahnya di waktu pohonnya berbuah dan (perhatikan pulalah) kematangannya. Sesungguhnya pada yang demikian itu ada tanda-tanda (kekuasaan Allah) bagi orang-orang yang beriman. (QS. Al-An'am: 99).

Menurut Tafsir Jalalayn: "(Dan Dialah yang menurunkan air hujan dari langit, lalu Kami tumbuhkan) dalam ayat ini terkandung iltifat dari orang yang ketiga menjadi pembicara (dengan air itu) yakni dengan air hujan itu (segala macam

tumbuh-tumbuhan) yang dapat tumbuh (maka Kami keluarkan darinya) dari tumbuh-tumbuhan itu sesuatu (tanaman yang hijau) yang menghijau (Kami keluarkan darinya) dari tanaman yang menghijau itu (butir yang banyak) yang satu sama lainnya bersusun seperti bulir-bulir gandum dan sejenisnya (dan dari pohon kurma) menjadi khabar dan dijadikan sebagai mubdal minhu (yaitu dari mayangnya) yaitu dari pucuk pohonnya; dan mubtadanya ialah (keluar tangkai-tangkainya) tunas-tunas buahnya (yang mengurai) saling berdekatan antara yang satu dengan yang lainnya (dan) Kami tumbuhkan berkat air hujan itu (kebun-kebun) tanaman-tanaman (anggur, zaitun dan delima yang serupa) dedaunannya; menjadi hal (dan yang tidak serupa) buahnya (perhatikanlah) hai orang-orang yang diajak bicara dengan perhatian yang disertai pemikiran dan pertimbangan (buahnya) dengan dibaca fathah huruf tsa dan huruf mimnya, atau dibaca dhammah keduanya sebagai kata jamak dari tsamrah; perihalnya sama dengan kata syajaratun jamaknya syajarun, dan khasyabatun jamaknya khasyabun (di waktu pohonnya berbuah) pada awal munculnya buah; bagaimana keadaannya? (dan) kepada (kematangannya) artinya kemasakannya, yaitu apabila telah masak; bagaimana keadaannya. (Sesungguhnya yang demikian itu ada tanda-tanda) yang menunjukkan kepada kekuasaan Allah swt. dalam menghidupkan kembali yang telah mati dan lain sebagainya (bagi orang-orang yang beriman) mereka disebut secara khusus sebab hanya merekalah yang dapat memanfaatkan hal ini untuk keimanan mereka, berbeda dengan orang-orang kafir".

Ayat tersebut memotivasi peneliti untuk menumbuhkan cabang / elemen baru proses bisnis yang dianalogikan seperti pohon. Dengan memberikan aturan dalam penerapan pertumbuhan diibaratkan dengan penggunaan air untuk menumbuhkan

suatu tanaman. Sesuai dengan yang Allah firmankan dalam Q.S Al-An'am bahwa "Dialah (Allah) yang menurunkan air hujan untuk menumbuhkan tanaman".

BAB 5

PENUTUP

5.1 Kesimpulan

Berdasarkan penelitian yang telah dilaksanakan, dapat disimpulkan bahwa :

1. Proses perhitungan *structural* dan *behavioral similarity* menggunakan empat algoritma yaitu: *Jaccard Coefficient Similarity*, *Overlap Coefficient Similarity*, *Dice Coefficient Similarity* dan *Cosine Coefficient Similarity*. Terbukti bahwa keempat algoritma tersebut dapat melakukan perhitungan mencari kemiripan model PNML pada proses bisnis Pondok Pesantren Salaf Anwarul Huda Malang (tipe pondok *salaf*), Pondok Pesantren Modern Ar-Rifaie 1 Gondanglegi (tipe pondok *modern*), Pondok Pesantren Mahasiswa Syai Urrifa' Malang (tipe pondok mahasiswa) dan Pondok Pesantren Ar-Rohmah Malang (tipe pondok *tahfidz*). Selanjutnya, dilakukan perhitungan *scalability* dengan menggunakan beberapa parameter yaitu: *similarity workflow*, jumlah elemen proses bisnis, *complexity* dan skala proses bisnis. Nilai *scalability* digunakan untuk untuk menentukan pola pertumbuhan proses bisnis. Pola pertumbuhan proses bisnis dilakukan dengan dua pendekatan yaitu: secara *sequence* dan percabangan.
2. Simulasi pertumbuhan proses bisnis dengan menerapkan teori *Production Rule Cellular Automata* untuk menumbuhkan beberapa elemen pada model A dengan pembanding model B.
3. Dilakukan pembobotan atau mengukur ketepatan pola pada saat proses simulasi pertumbuhan pada dua pendekatan percabangan dan *sequence*. Hasil dari kedua

pembobotan akan berpengaruh pada iterasi yang dihasilkan. Meskipun dilakukan berulang kali percobaan simulasi pertumbuhan dengan masing-masing pembobotan, hasil simulasi akan terus berbeda. Hal ini dikarenakan pertumbuhan elemen pada proses bisnis dilakukan secara *random*.

4. Simulasi pertumbuhan proses bisnis dari segi *scalability* terbukti berhasil karena adanya penurunan nilai *scalability* dan perhitungan *standart* deviasi menghasilkan rata-rata dengan nilai "<0.2".
5. Simulasi pertumbuhan proses bisnis dari segi *similarity*, hasil yang didapatkan belum maksimal karena adanya pertumbuhan *random*, sehingga rata-rata nilai *similarity* setelah simulasi "<0.5". Terbukti berhasil apabila nilai similarity setelah simulasi > sebelum simulasi pertumbuhan.

5.2 Saran

Berikut ini merupakan beberapa saran untuk penelitian di masa akan datang. Saran-saran ini didasarkan pada hasil perancangan, implementasi dan pengujian pada sistem. Saran-saran tersebut antara lain :

1. Data masukan sistem untuk perhitungan *similarity* tidak hanya berupa *file* pemodelan PNML, melainkan dapat ditambah dengan *file* pemodelan yang lain.
2. Memasukkan faktor eksternal dalam simulasi pertumbuhan proses bisnis pada penelitian yang akan datang.
3. Memodelkan simulasi pertumbuhan dengan lebih formal artinya dengan beberapa metode yang dirasa lebih baik. Contoh: metode *L-System*.
4. Metode yang digunakan dapat dimasukkan dalam konfigurasi sistem yang akan dibangun.

5. Pada penelitian selanjutnya, diharapkan tidak menumbuhkan proses bisnis secara *random*, karena hasil yang didapatkan dari segi nilai *similarity* belum maksimal.

DAFTAR PUSTAKA

Ainul Muhammad [et al.] Scalability Measurement of Business Process Model Using Business Processes Similarity and Complexity [Journal]. - 2016.

Dewanto Wawan and Falalah Menyelaraskan Teknologi dengan Strategi Bisnis [Book Section] // ERP (Enterprise Resource Planning). - [s.l.] : Informatika Bandung.

Essam Marwa M. and Mansar Selma Limam Towards a Software Framework for Automatic Business Process Redesign [Journal]. - Egypt : Ain Shams University, 2011.

Fajarivan Muhammad Pratama Integrasi Sistem Keuangan pada Enterprise Resource Planning Pondok Pesantren Tipe Pondok D Menggunakan Services Oriented Architecture [Report]. - Malang : [s.n.], 2017.

Harmon Paul and Wolf Celia The State of the BPM Market [Book]. - [s.l.] : BPTrends Report, 2014.

Hopcroft, E John and Ulman Jeffrey D Introduction to Automata Theory, Language and Computation [Book]. - [s.l.] : Addison-Wesley, 2006.

Kamus Besar Bahasa Indonesia [Online]. - kbbi.web.id/simulasi.

Peterson James L Petri Net Theory and the Modelling of System [Book]. - United States of America : Prentice Hall Inc, 1981.

Prasetya Arif Wahyu Aplikasi Pengolahan Proses Bisnis Menggunakan Metode Analisis Behavioral, Structural dan Semantic untuk Meningkatkan Akurasi dalam Penentuan Common Fragment Workflow pada ERP Pondok Pesantren [Report]. - Malang : [s.n.], 2017.

Puspa Dewi Lily, Indahyati Uce and Hari S Yulius Pemodelan Proses Bisnis Menggunakan Activity Diagram UML dan BPMN [Journal]. - 2010.

Rini Anggrainingsih, dkk Analsisis Dan Verifikasi Workflow Menggunakan Petri (Studi kasus: Proses Bisnis di Universitas Sebelas Maret) [Conference] // Senimar Nasional Teknologi Informasi & Komunikasi Terapan 2014. - 2014.

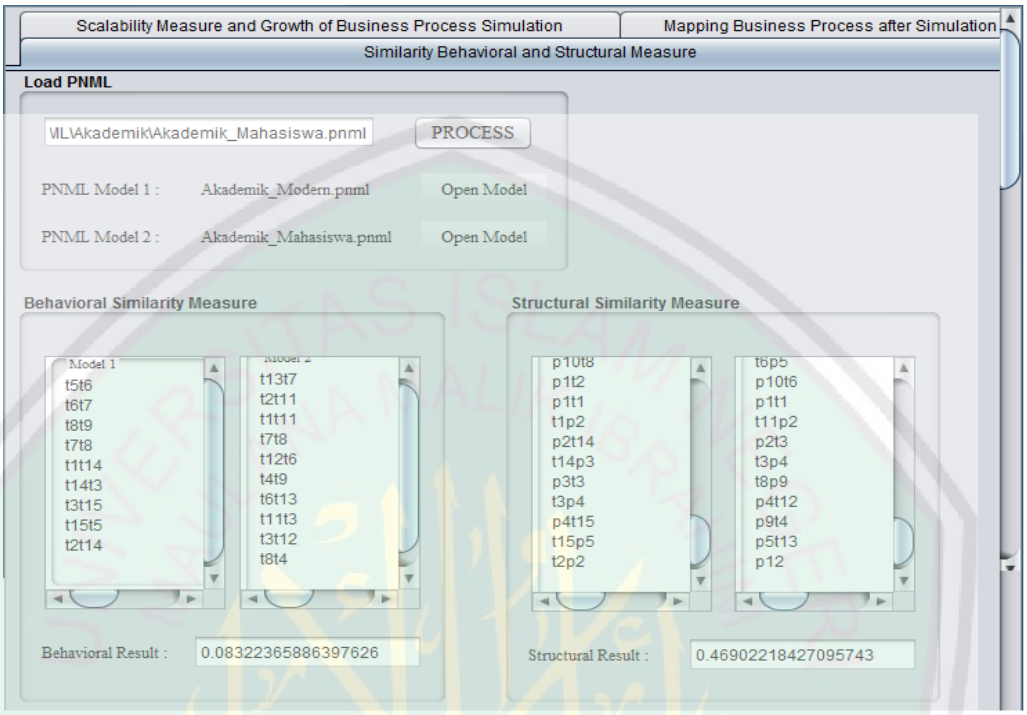
Rolón Elvira [et al.] Analysis and Validation of Control-Flow Complexity Measures with BPMN Process Models [Book Section] // Enterprise, Business Process and Information Systems Modeling. - Amsterdam : Springer Berlin Heidelberg, 2009.

Suhartono Intergration of Artifial Neural Network Into Genetic L-System Programming Based Plant Modeling Environment with Mathematica [Book]. - Jakarta Pusat : Kementrian Agama Republik Indonesia, 2012.

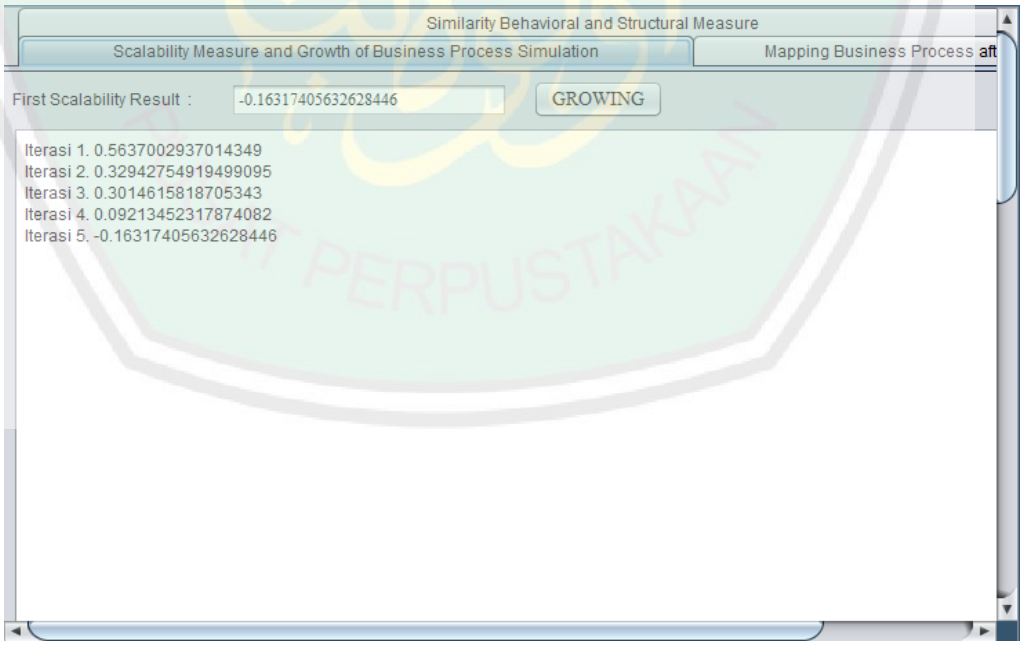
Wijaya Santo F and Darudito Suparto ERP (Enterprise Resource Planning) & Solusi Bisnis [Book]. - Yogyakarta : Graha Ilmu, 2009.

LAMPIRAN

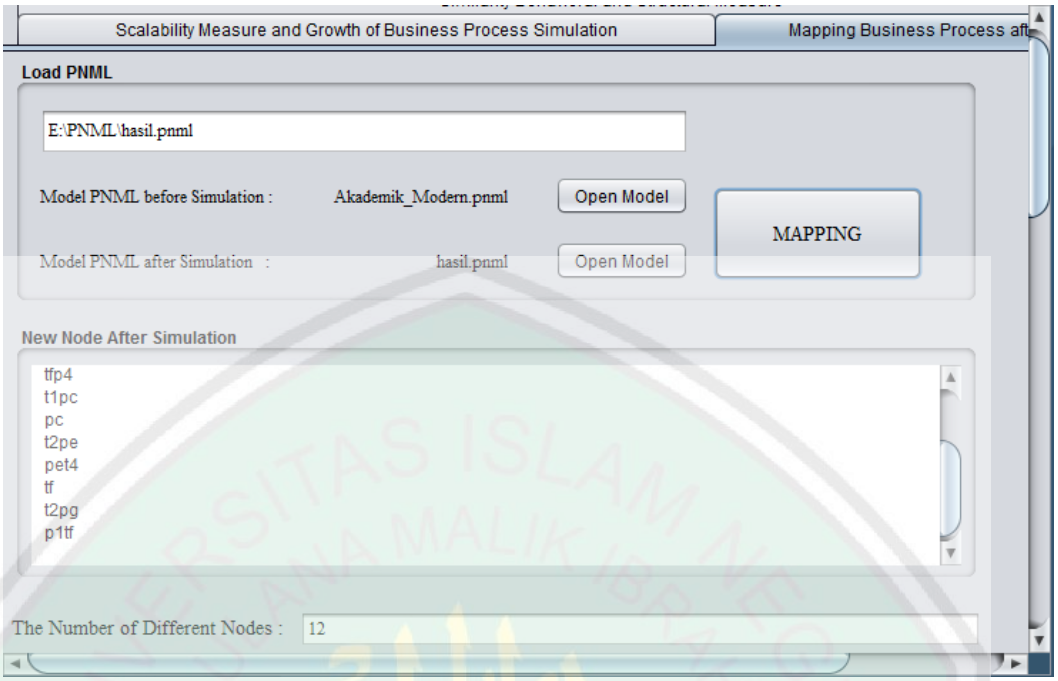
BAGIAN AKADEMIK



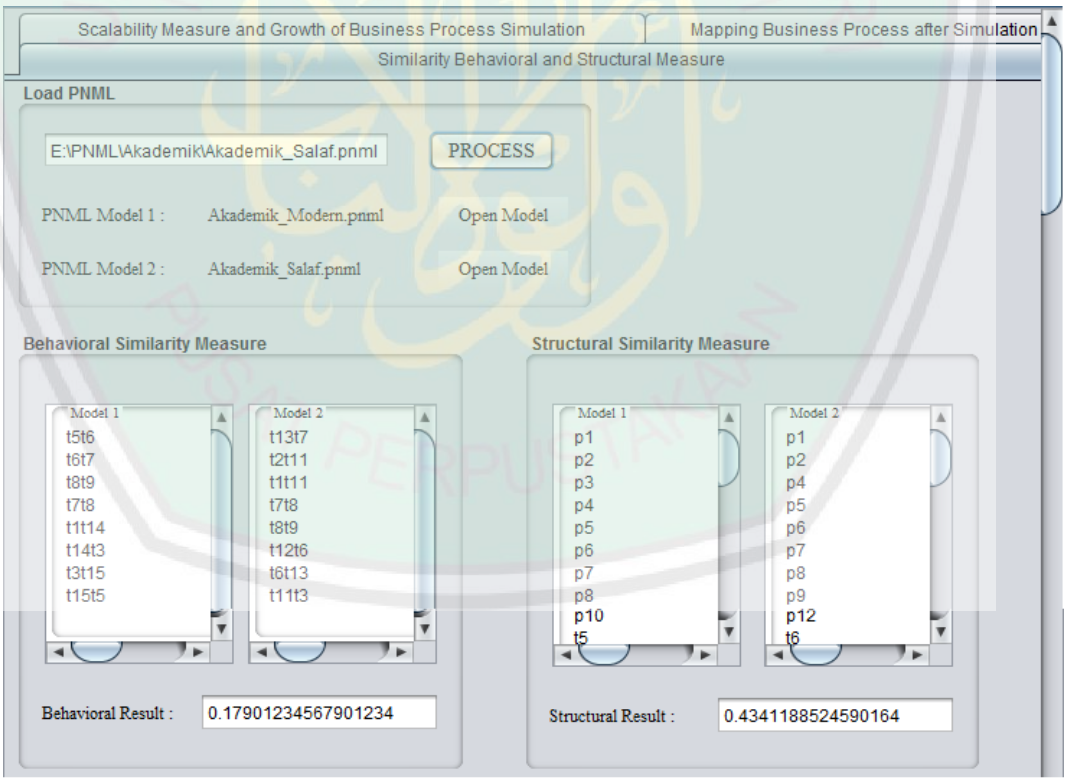
Gambar A 1 Perhitungan Similarity antar model Akademik *Modern* dan Akademik Mahasiswa



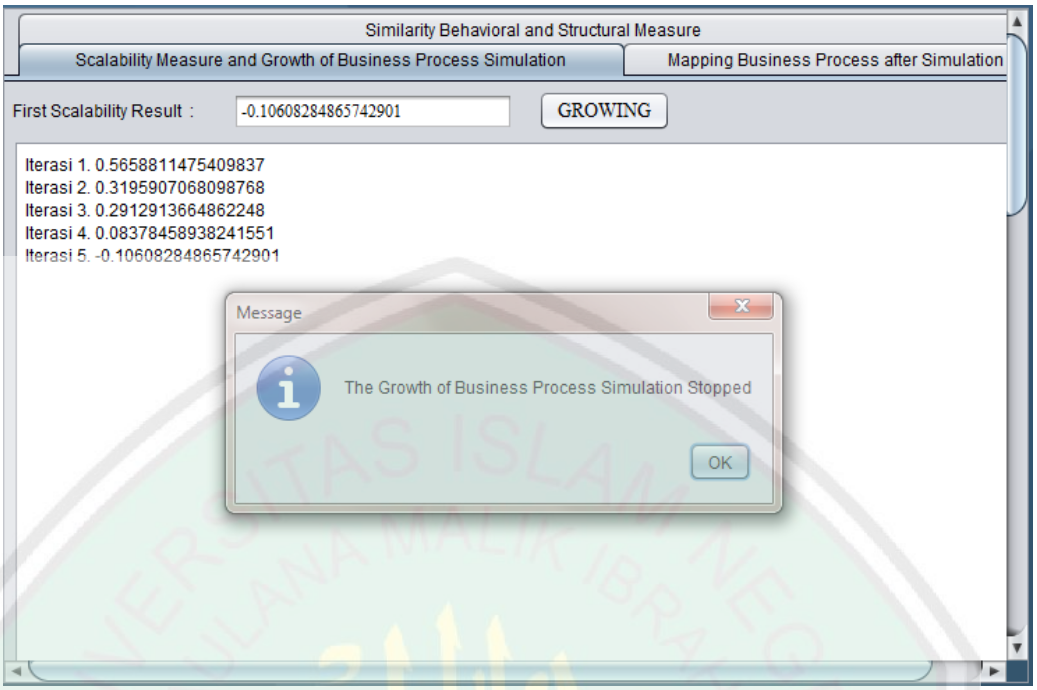
Gambar A 2 Hasil Simulasi Pertumbuhan model antar Akademik *Modern* dan Akademik Mahasiswa



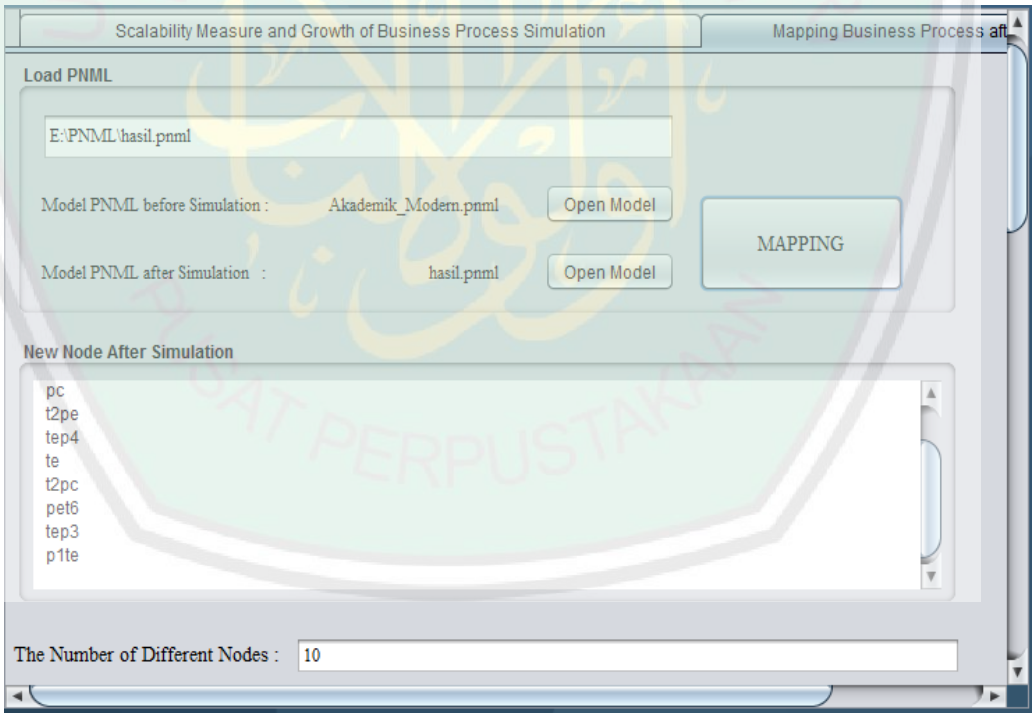
Gambar A 3 *Mapping* model antar Akademik *Modern* dan hasil dari Simulasi model Akademik *Modern*-Akademik Mahasiswa



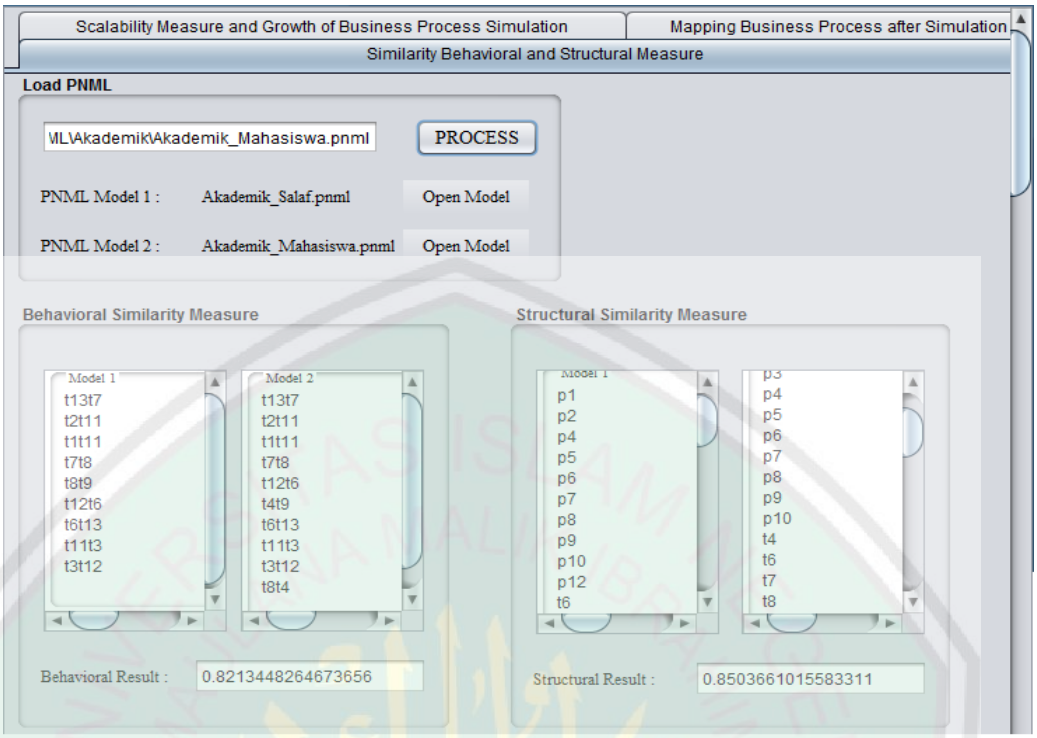
Gambar A 4 Perhitungan *Similarity* antar model Akademik *Modern* dan Akademik *Salaf*



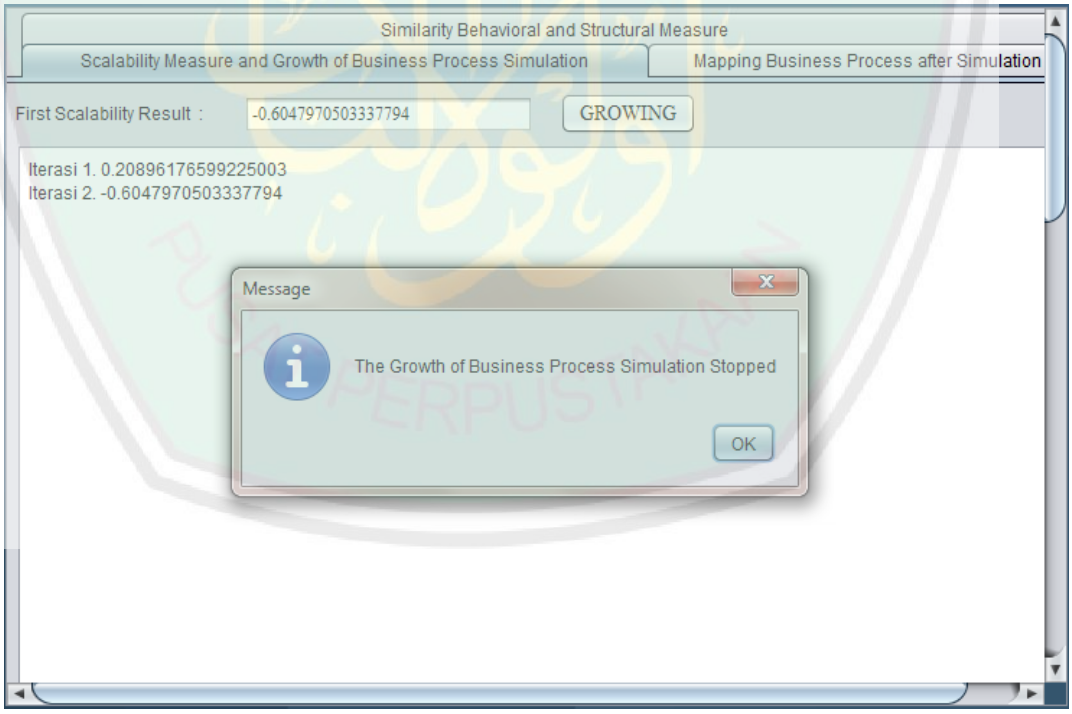
Gambar A 5 Hasil Simulasi Pertumbuhan model antar Akademik *Modern* dan Akademik *Salaf*



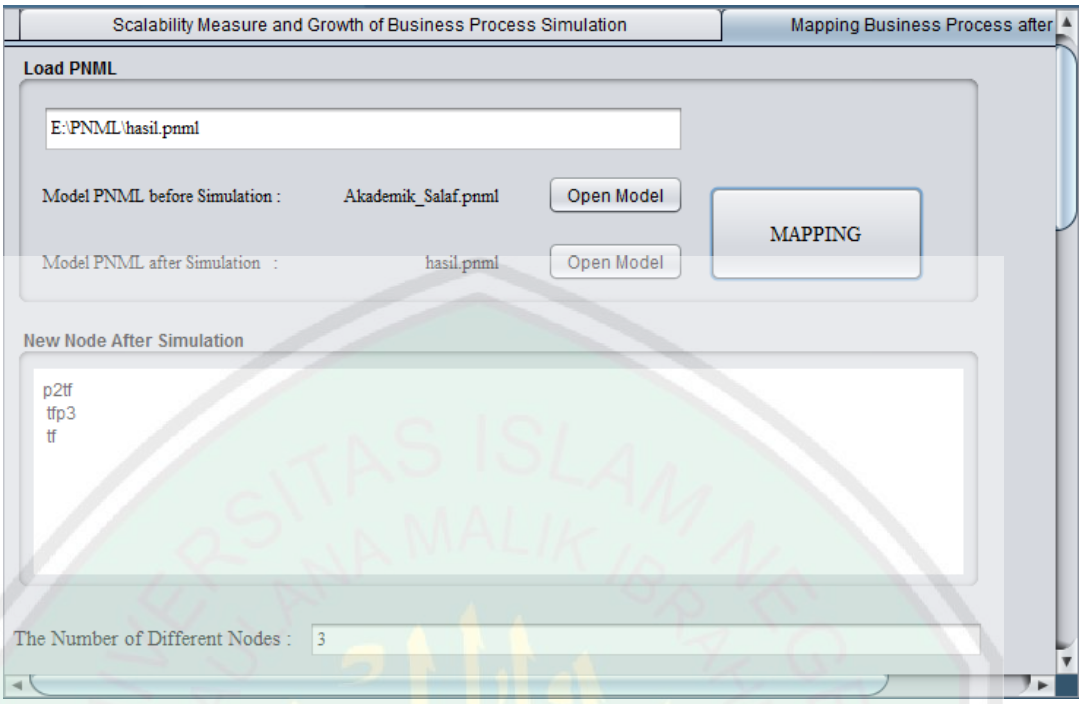
Gambar A 6 *Mapping* model antar Akademik *Modern* dan hasil dari Simulasi model Akademik *Modern*-Akademik *Salaf*



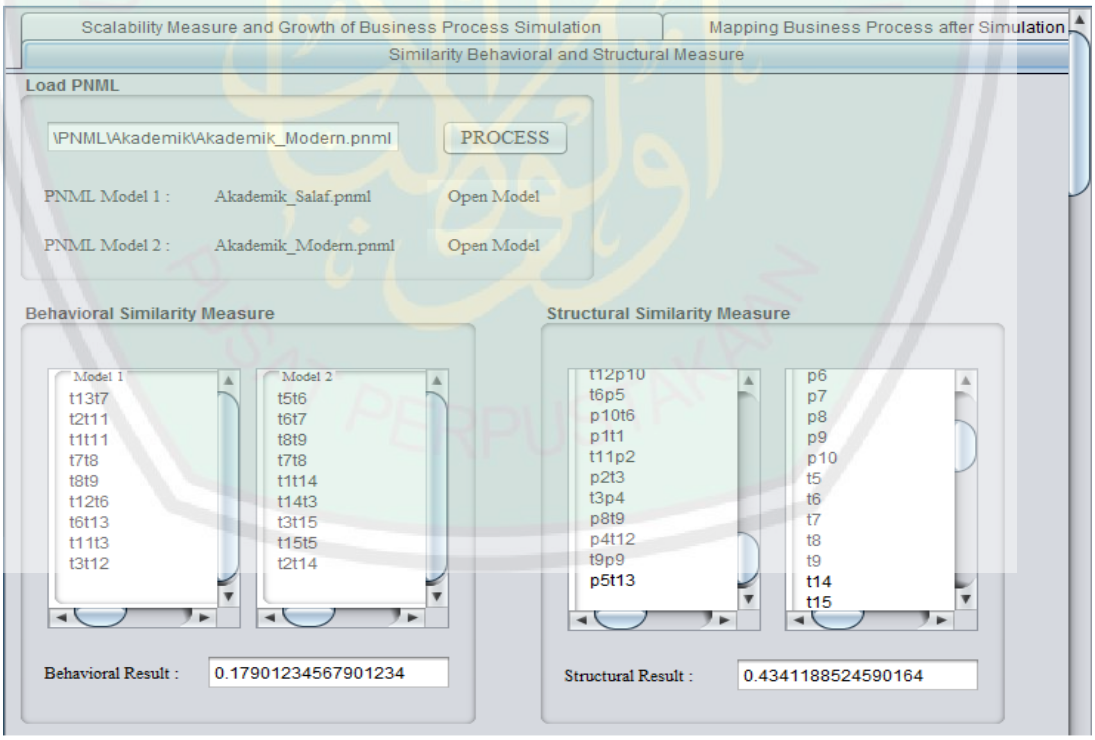
Gambar A 7 Perhitungan *Similarity* antar model Akademik *Salaf* dan Akademik Mahasiswa



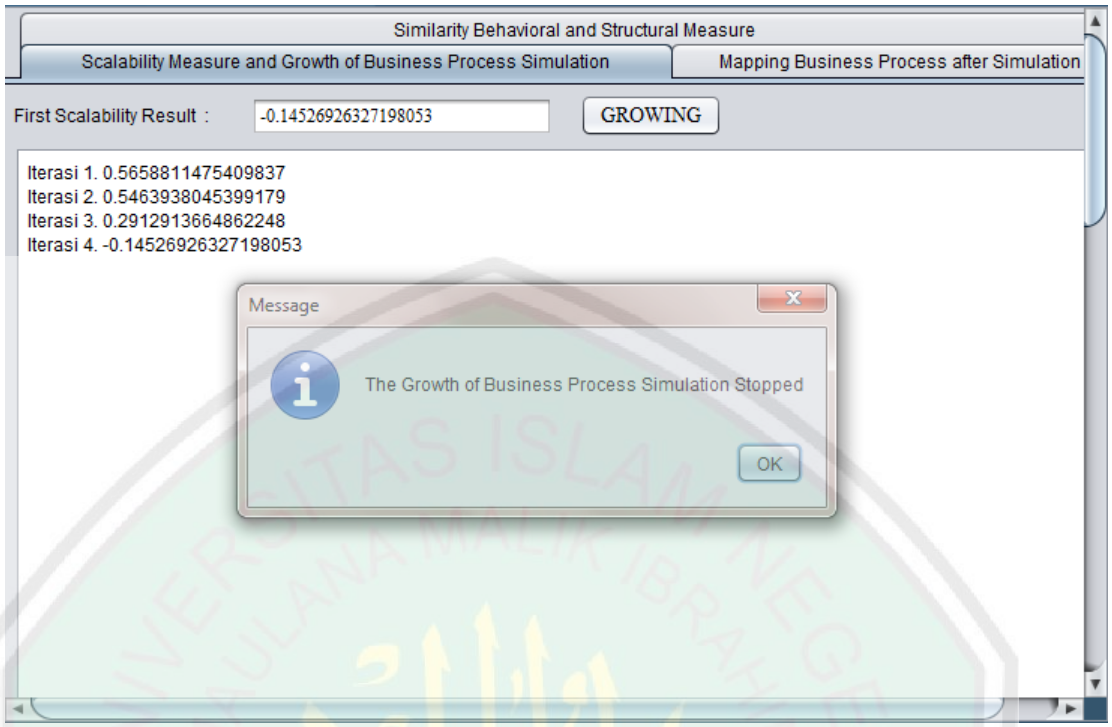
Gambar A 8 Hasil Simulasi Pertumbuhan model antar Akademik *Salaf* dan Akademik Mahasiswa



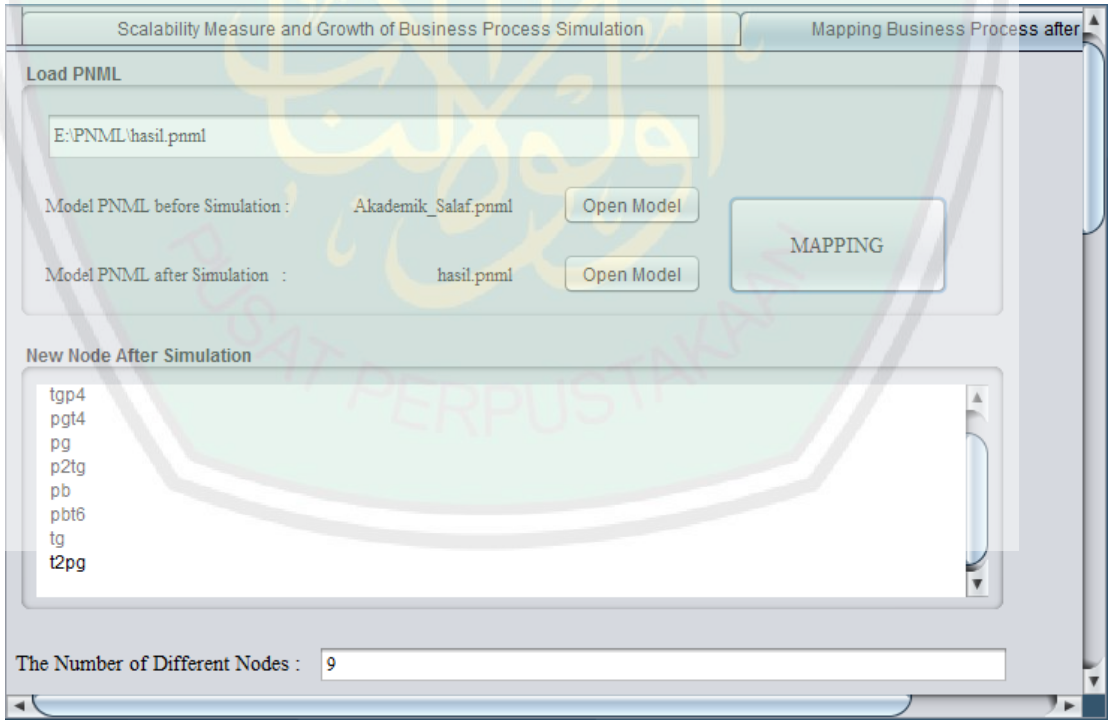
Gambar A 9 *Mapping* model antar Akademik *Salaf* dan hasil dari Simulasi model Akademik *Salaf* -Akademik Mahasiswa



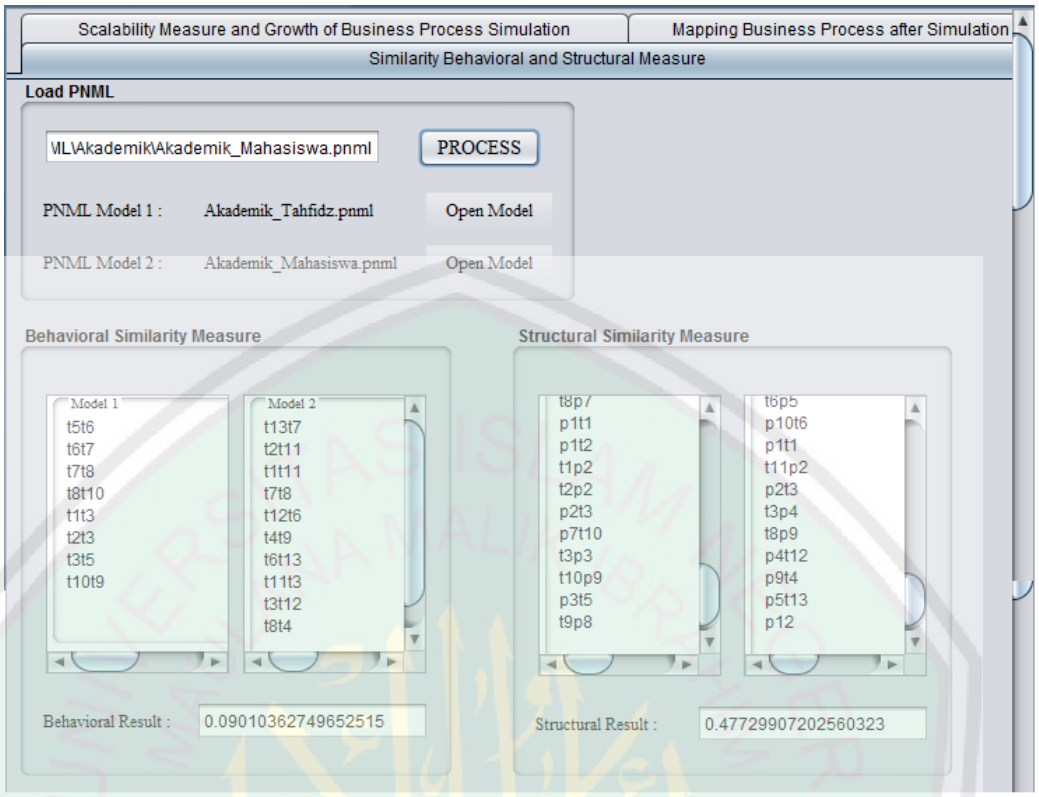
Gambar A 10 Perhitungan *Similarity* antar model Akademik *Salaf* dan Akademik *Modern*



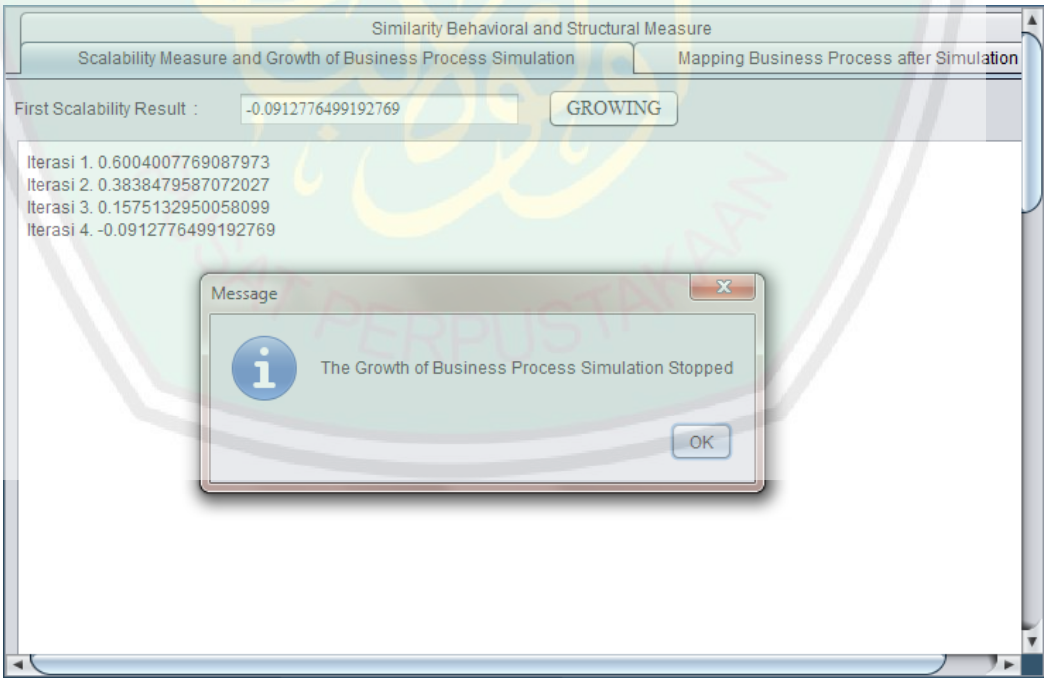
Gambar A 11 Hasil Simulasi Pertumbuhan model antar Akademik *Salaf* dan Akademik *Modern*



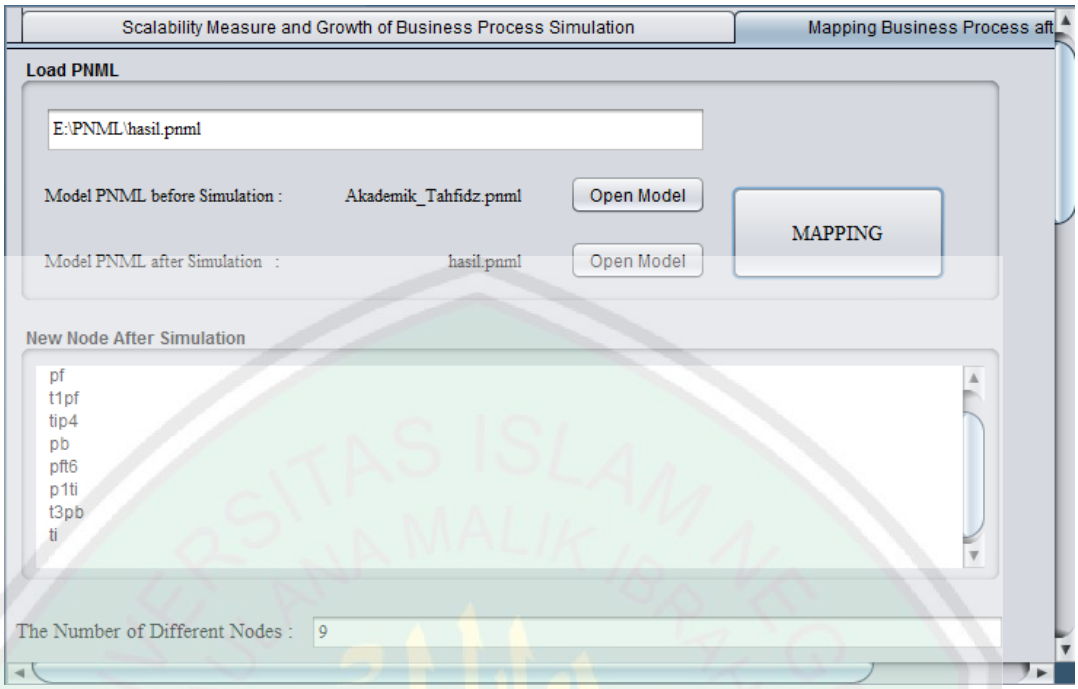
Gambar A 12 *Mapping* model antar Akademik *Salaf* dan hasil dari Simulasi model Akademik *Salaf* -Akademik *Modern*



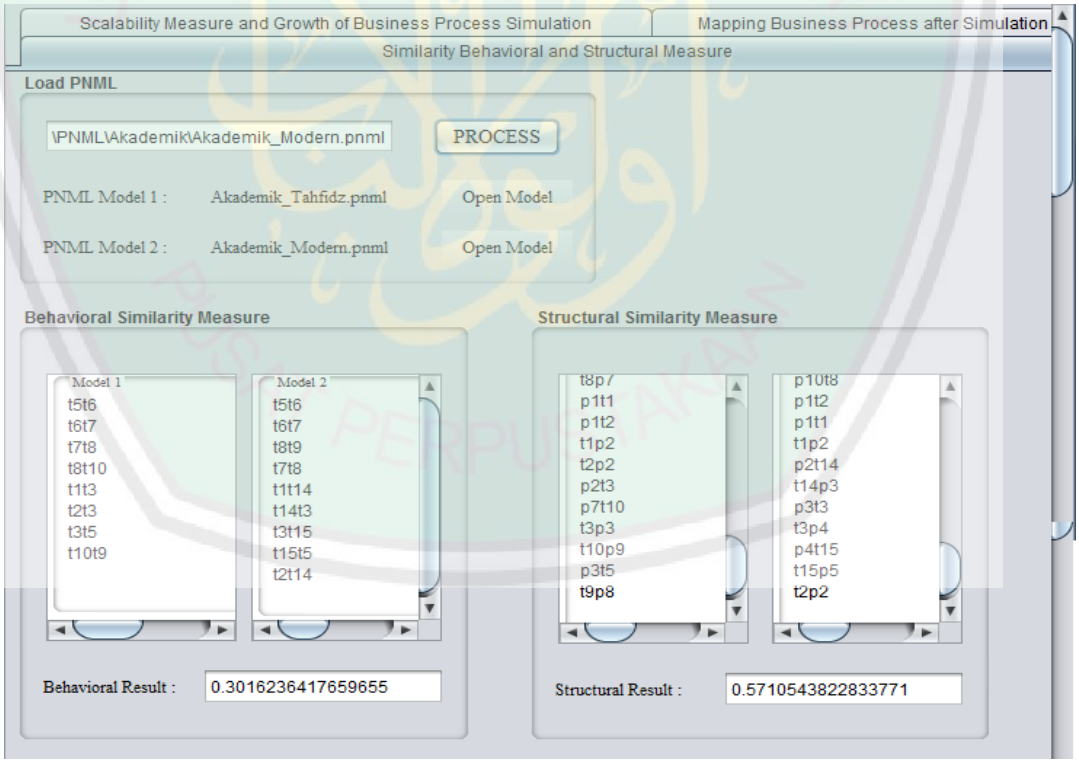
Gambar A 13 Perhitungan *Similarity* antar model Akademik *Tahfidz* dan Akademik Mahasiswa



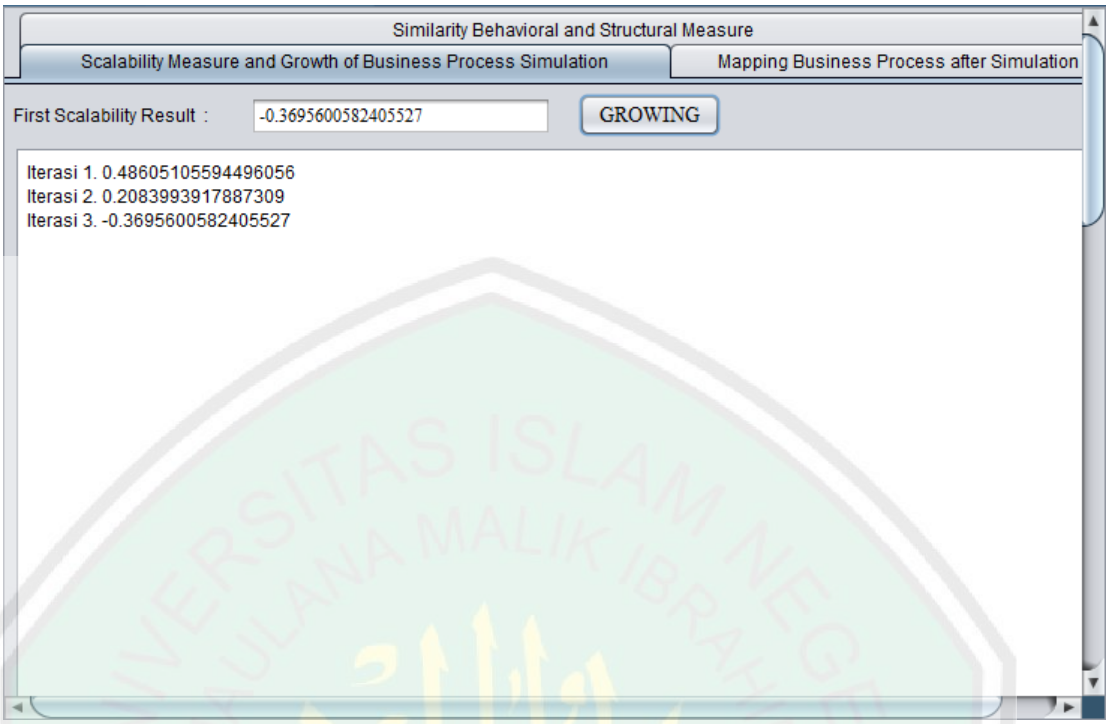
Gambar A 14 Hasil Simulasi Pertumbuhan model antar Akademik *Tahfidz* dan Akademik Mahasiswa



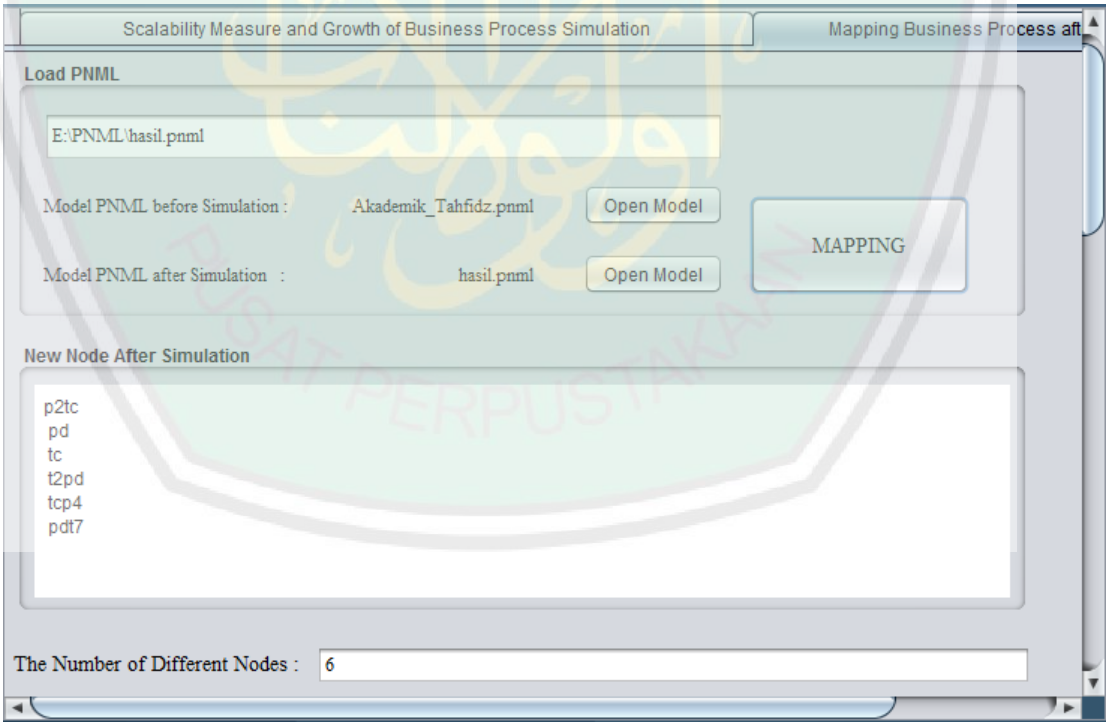
Gambar A 15 *Mapping* model antar Akademik *Tahfidz* dan hasil dari Simulasi model Akademik *Tahfidz* -Akademik Mahasiswa



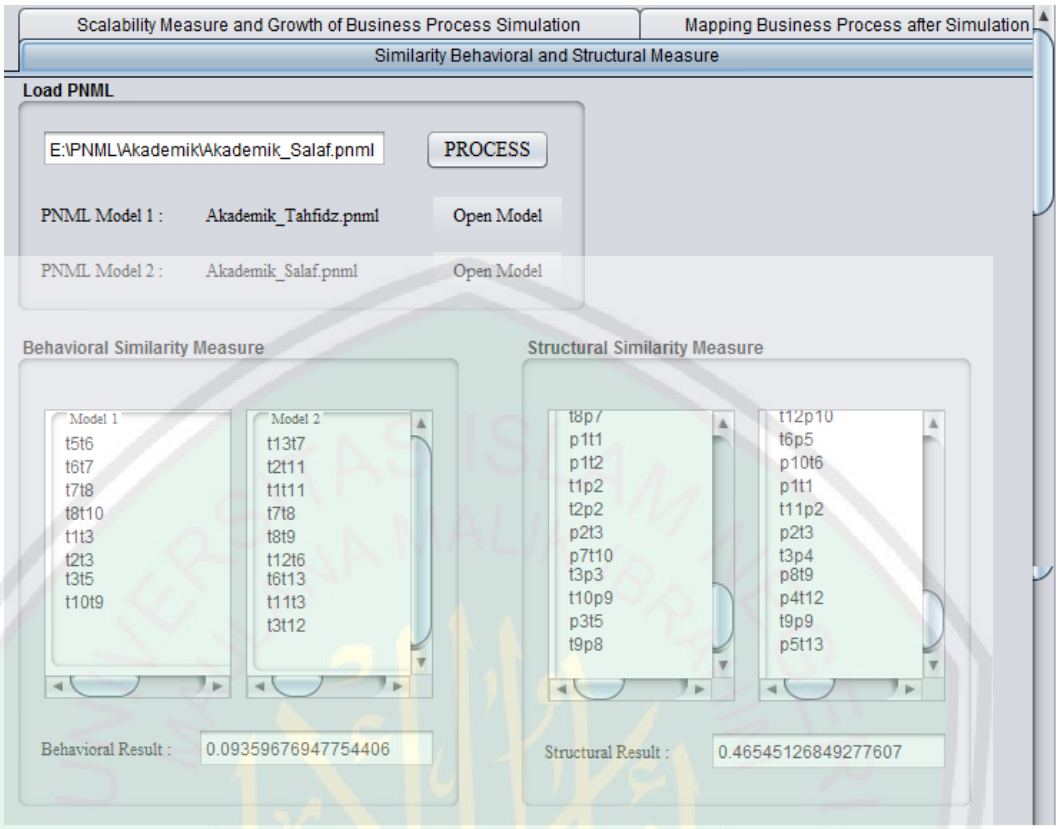
Gambar A 16 Perhitungan *Similarity* antar model Akademik *Tahfidz* dan Akademik *Modern*



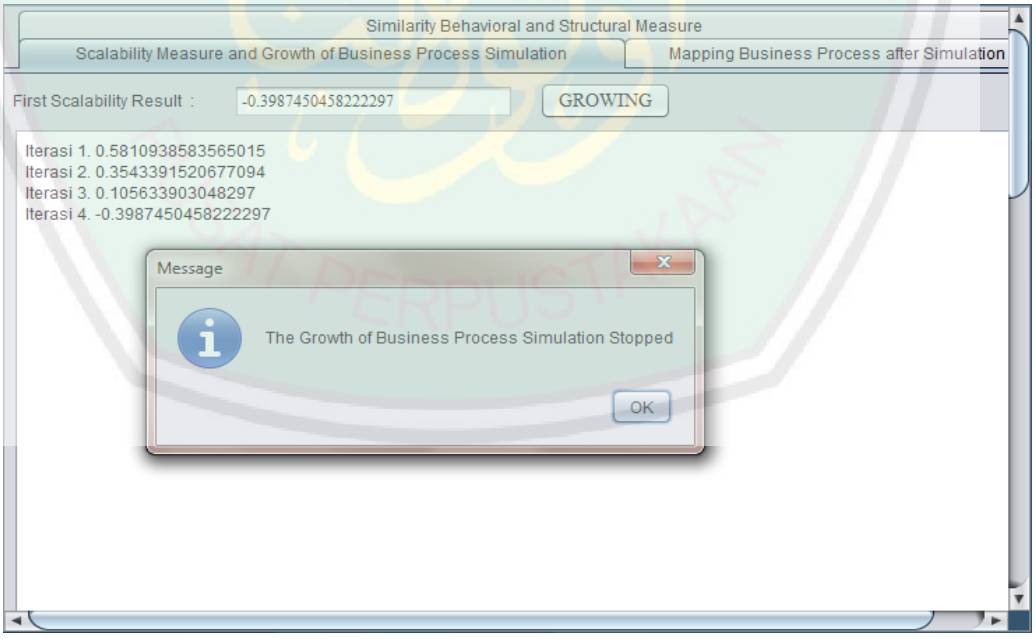
Gambar A 17 Hasil Simulasi Pertumbuhan model antar Akademik *Tahfidz* dan Akademik *Modern*



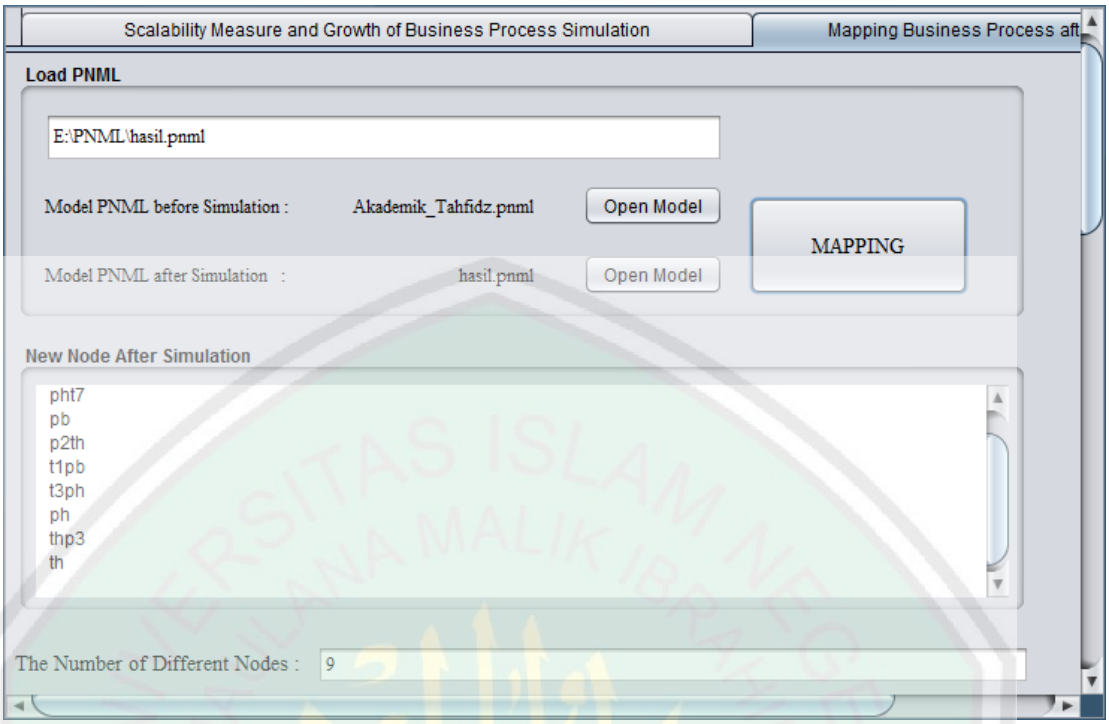
Gambar A 18 *Mapping* model antar Akademik *Tahfidz* dan hasil dari Simulasi model Akademik *Tahfidz* -Akademik *Modern*



Gambar A 19 Perhitungan *Similarity* antar model Akademik *Tahfidz* dan Akademik *Salaf*

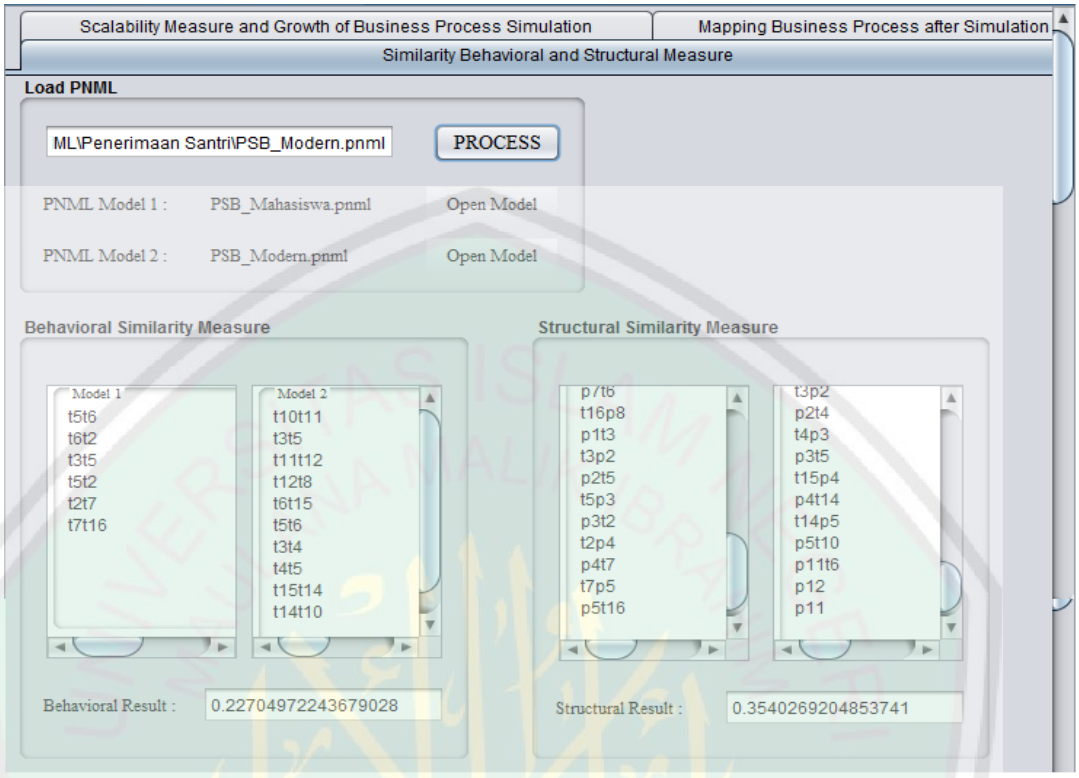


Gambar A 20 Hasil Simulasi Pertumbuhan model antar Akademik *Tahfidz* dan Akademik *Salaf*

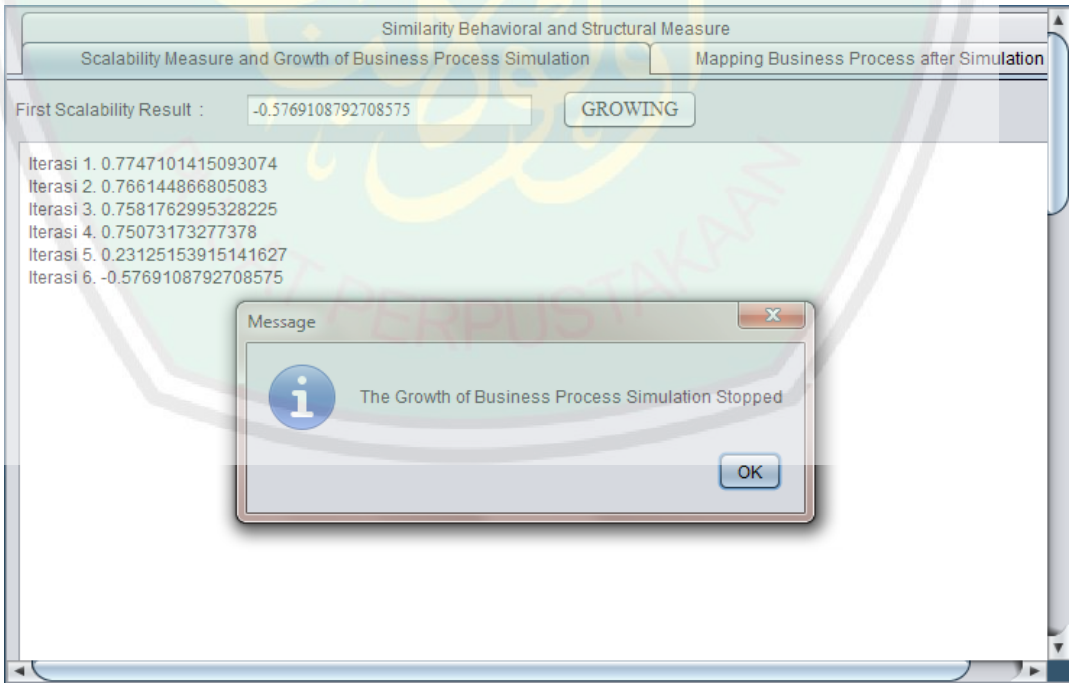


Gambar A 21 *Mapping* model antar Akademik *Tahfidz* dan hasil dari Simulasi model Akademik *Tahfidz* -Akademik *Salaf*

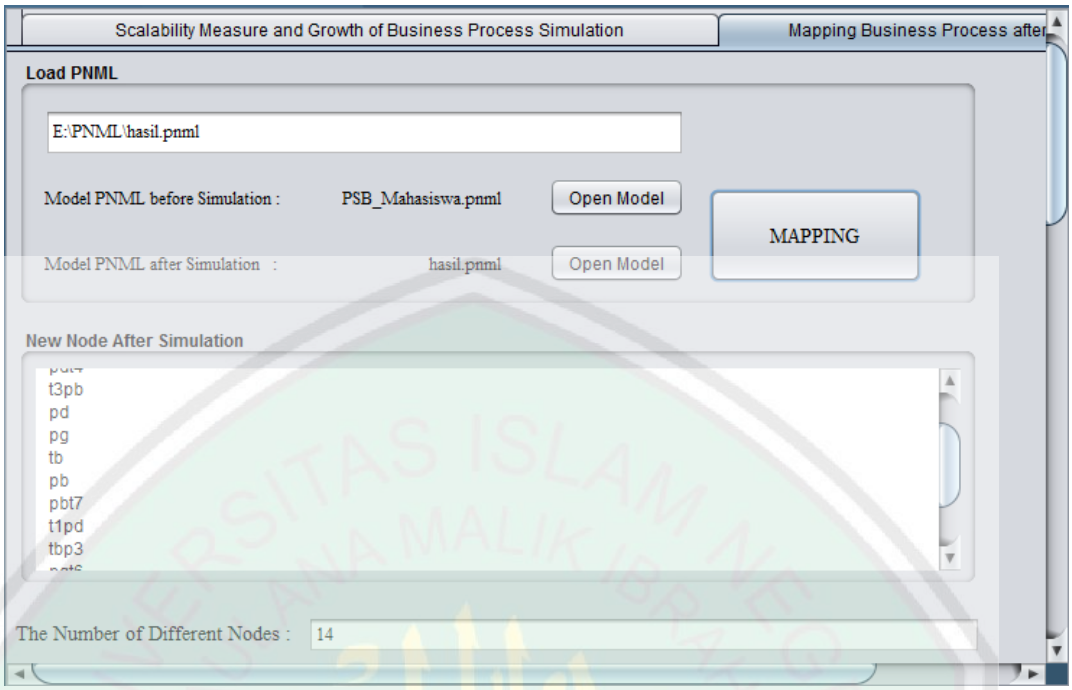
BAGIAN PSB



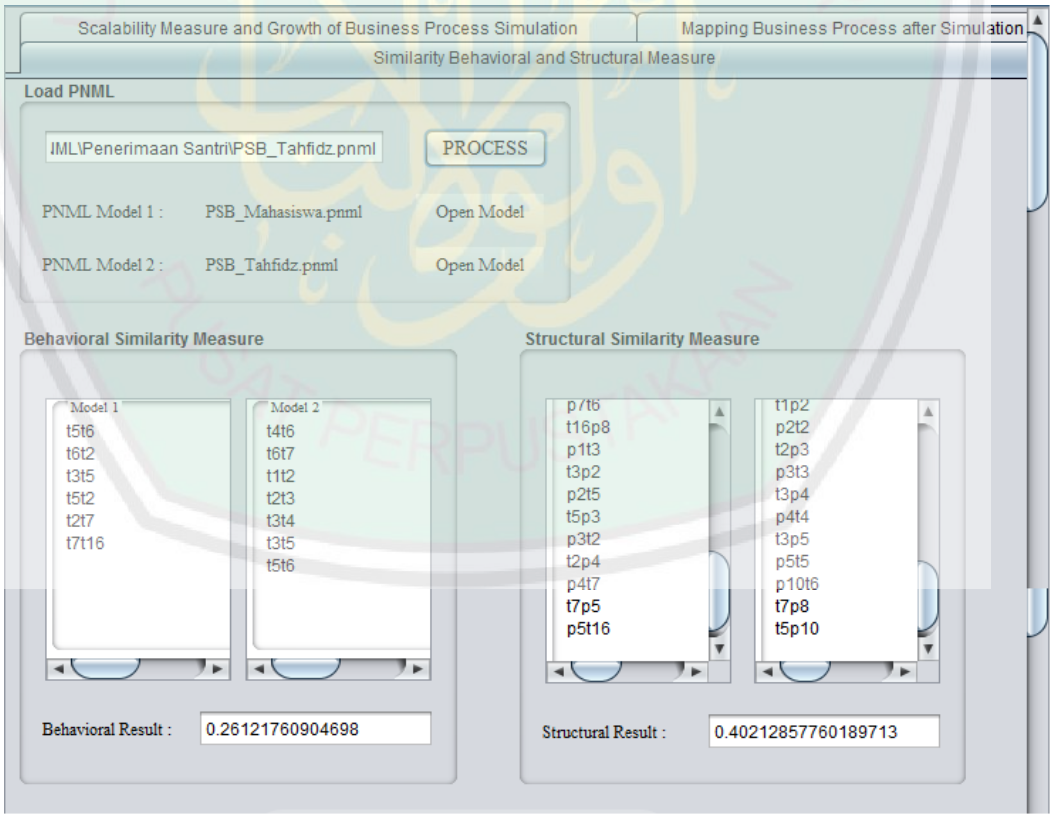
Gambar A 22 Perhitungan *Similarity* antar model PSB Mahasiswa dan PSB *Modern*



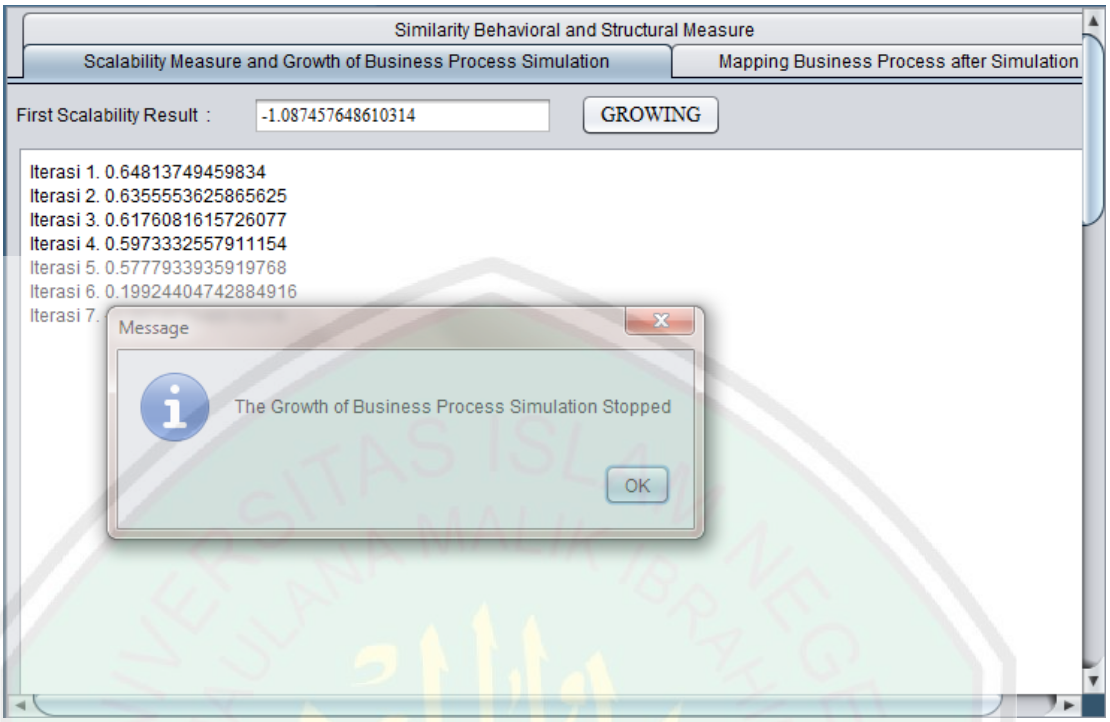
Gambar A 23 Hasil Simulasi Pertumbuhan model antar PSB Mahasiswa dan PSB *Modern*



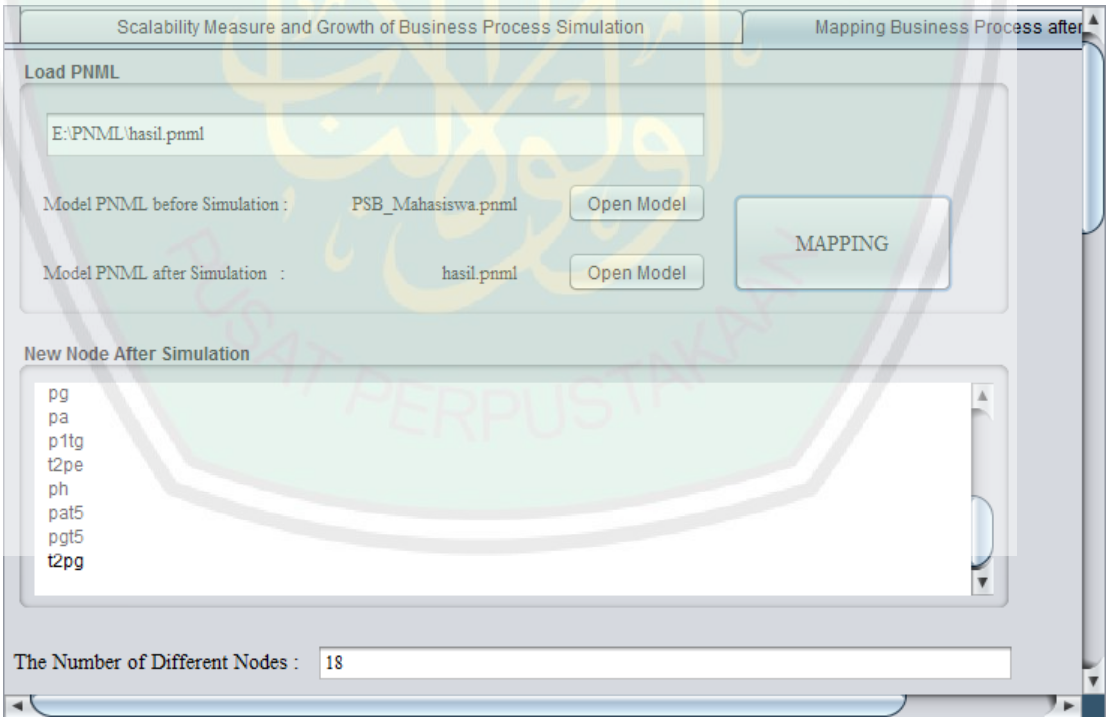
Gambar A 24 *Mapping* model antar PSB Mahasiswa dan hasil dari Simulasi model PSB Mahasiswa-PSB *Modern*



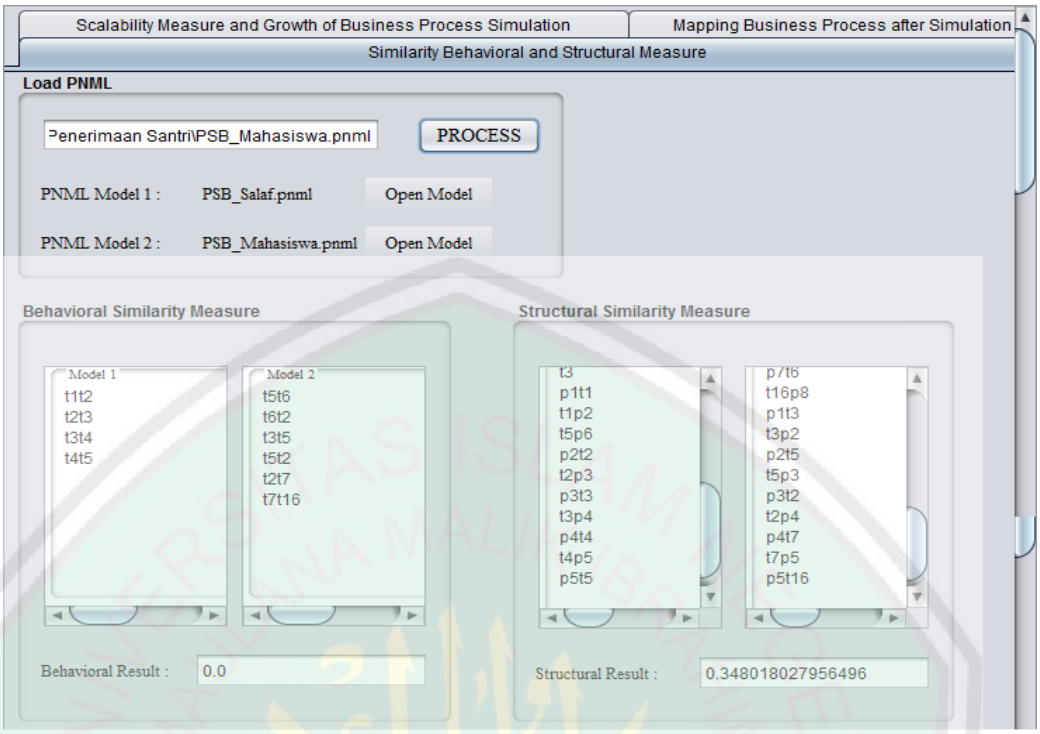
Gambar A 25 Perhitungan *Similarity* antar model PSB Mahasiswa dan PSB *Tahfidz*



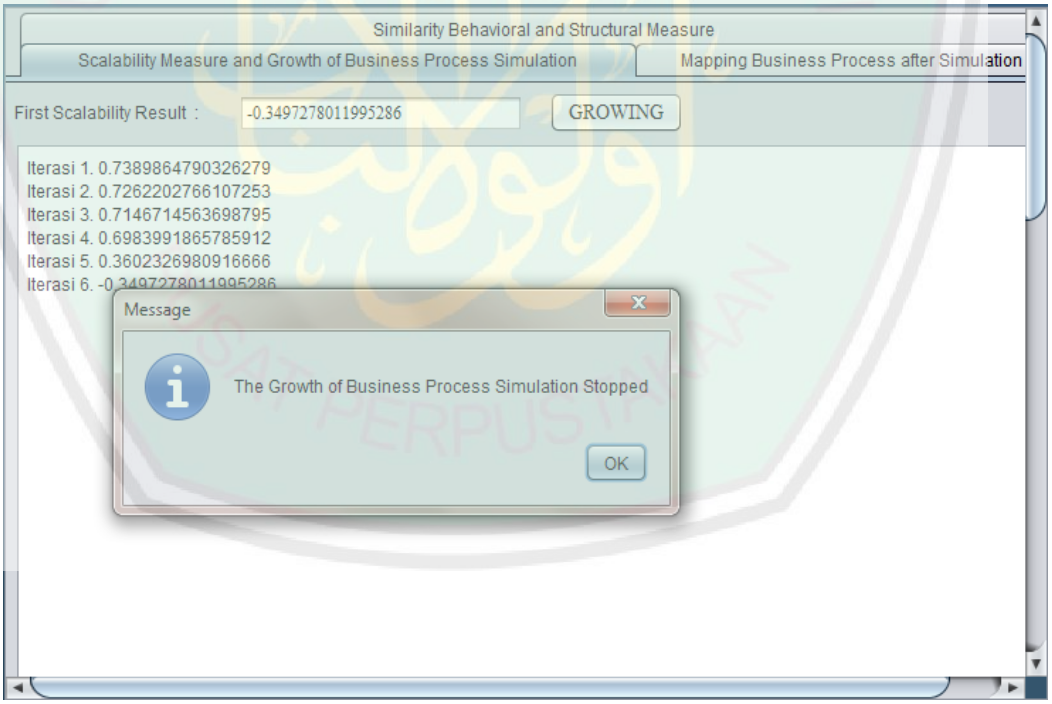
Gambar A 26 Hasil Simulasi Pertumbuhan model antar PSB Mahasiswa dan PSB *Tahfidz*



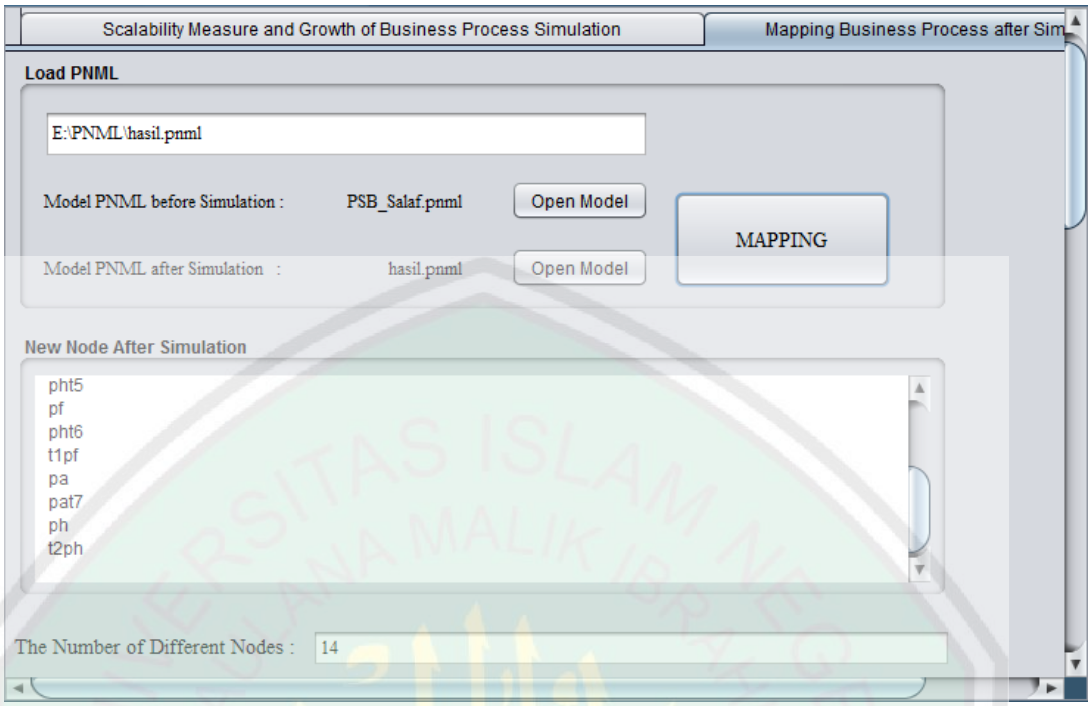
Gambar A 27 *Mapping* model antar PSB Mahasiswa dan hasil dari Simulasi model PSB Mahasiswa-PSB *Tahfidz*



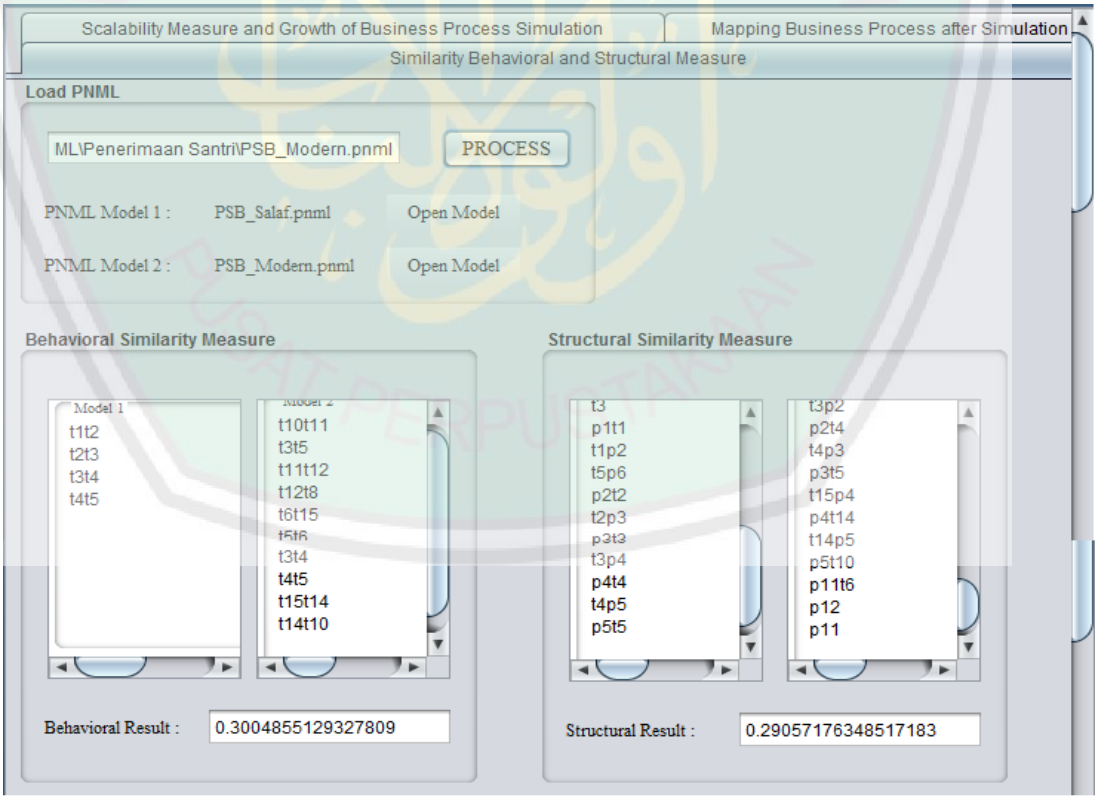
Gambar A 28 Perhitungan *Similarity* antar model PSB *Salaf* dan PSB Mahasiswa



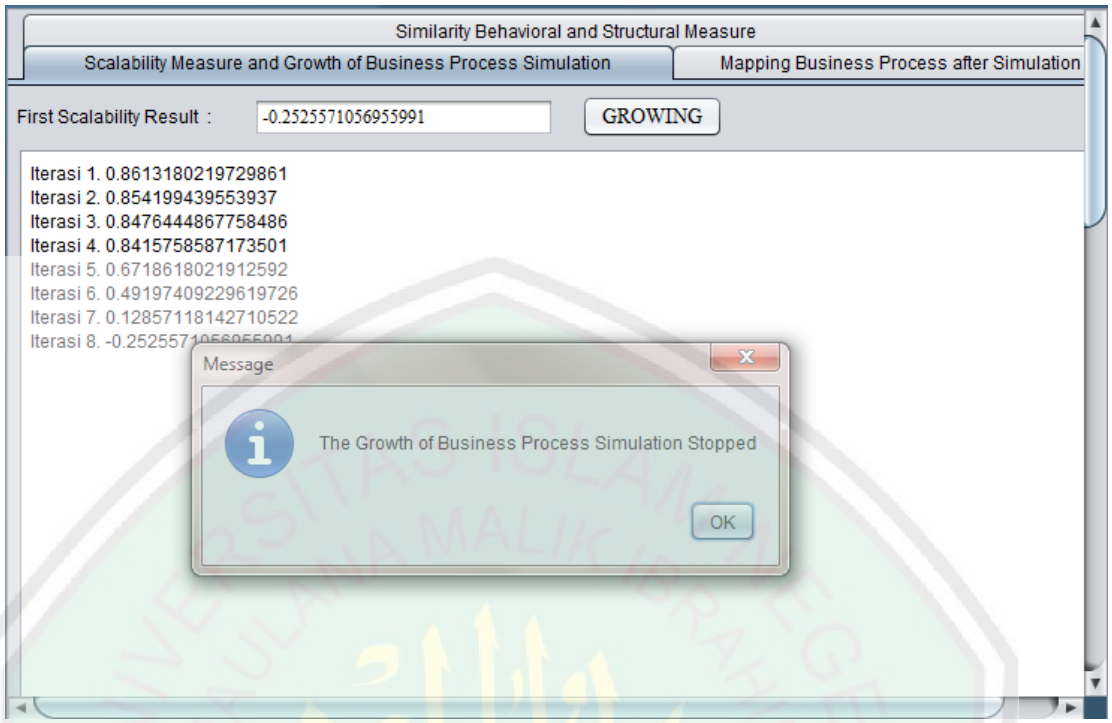
Gambar A 29 Hasil Simulasi Pertumbuhan model antar PSB *Salaf* dan PSB Mahasiswa



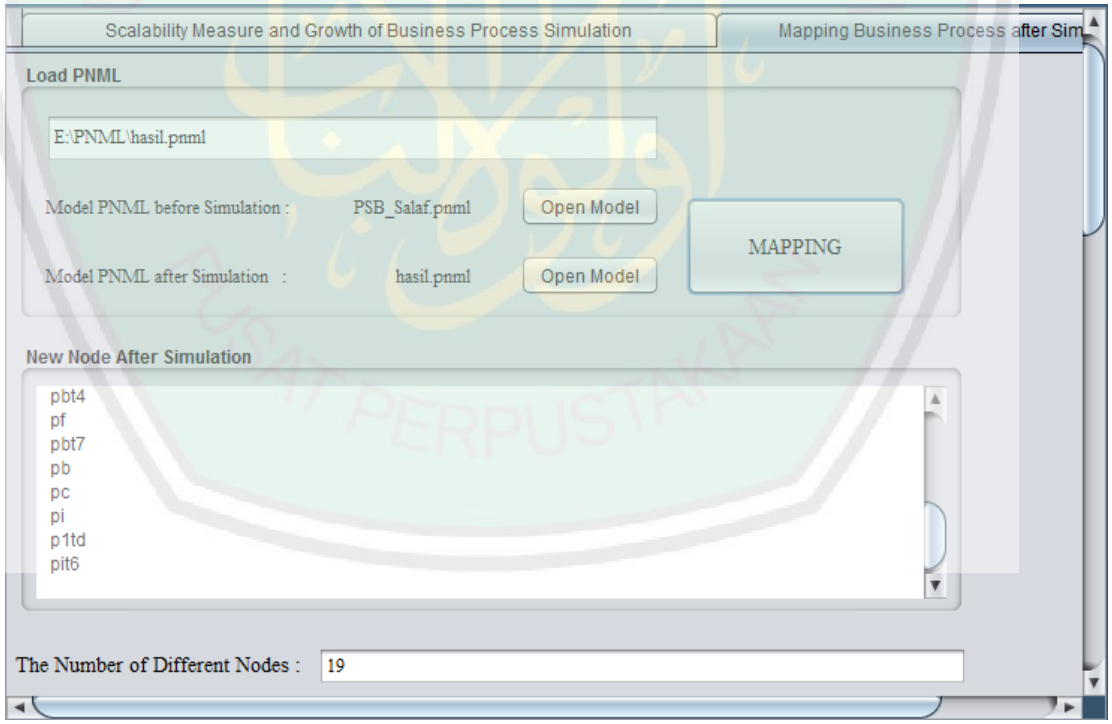
Gambar A 30 *Mapping* model antar PSB *Salaf* dan hasil dari Simulasi model PSB *Salaf* - PSB Mahasiswa



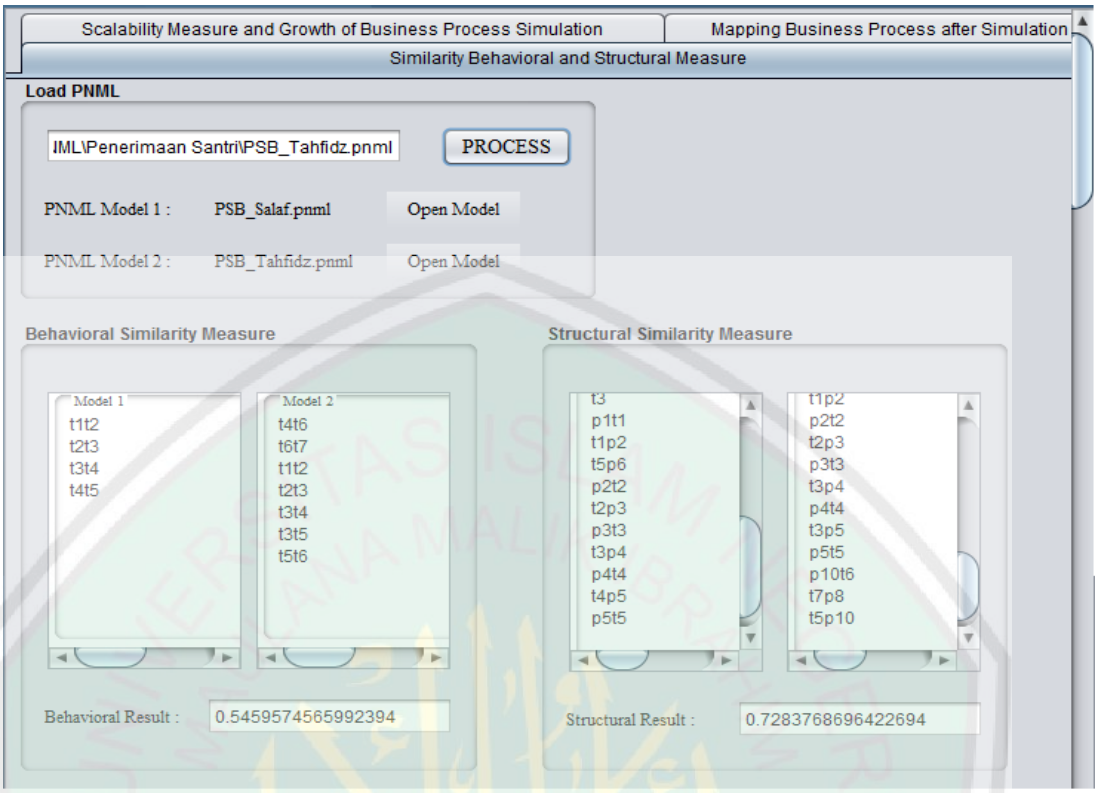
Gambar A 31 Perhitungan *Similarity* antar model PSB *Salaf* dan PSB *Modern*



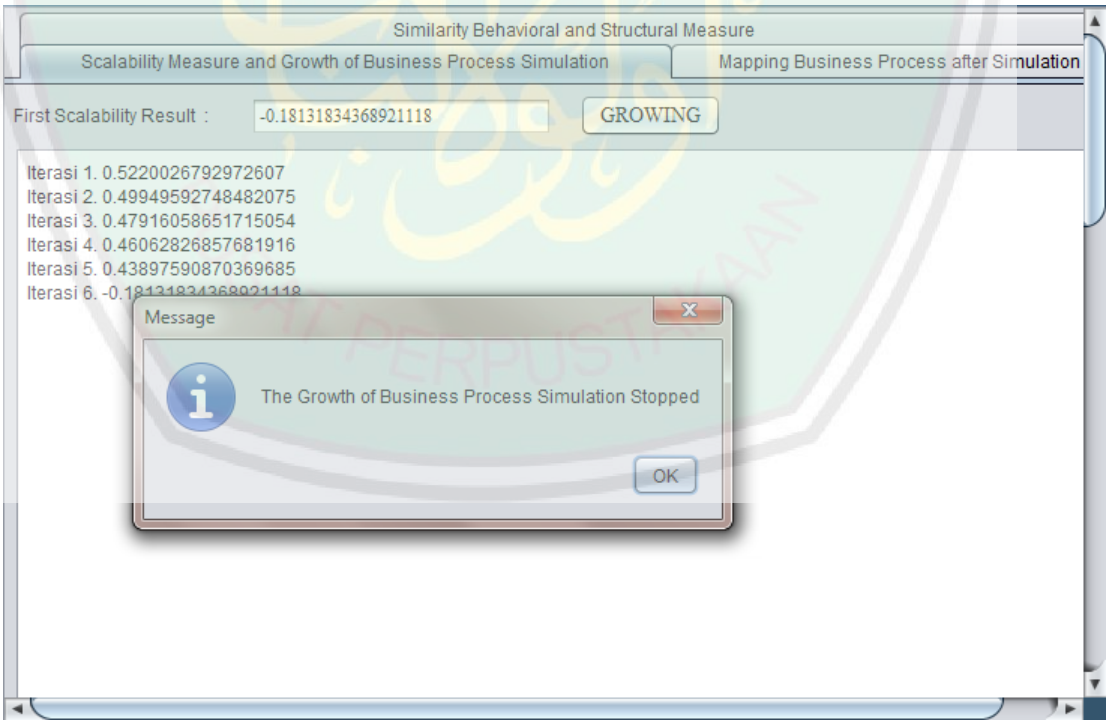
Gambar A 32 Hasil Simulasi Pertumbuhan model antar PSB *Salaf* dan PSB *Modern*



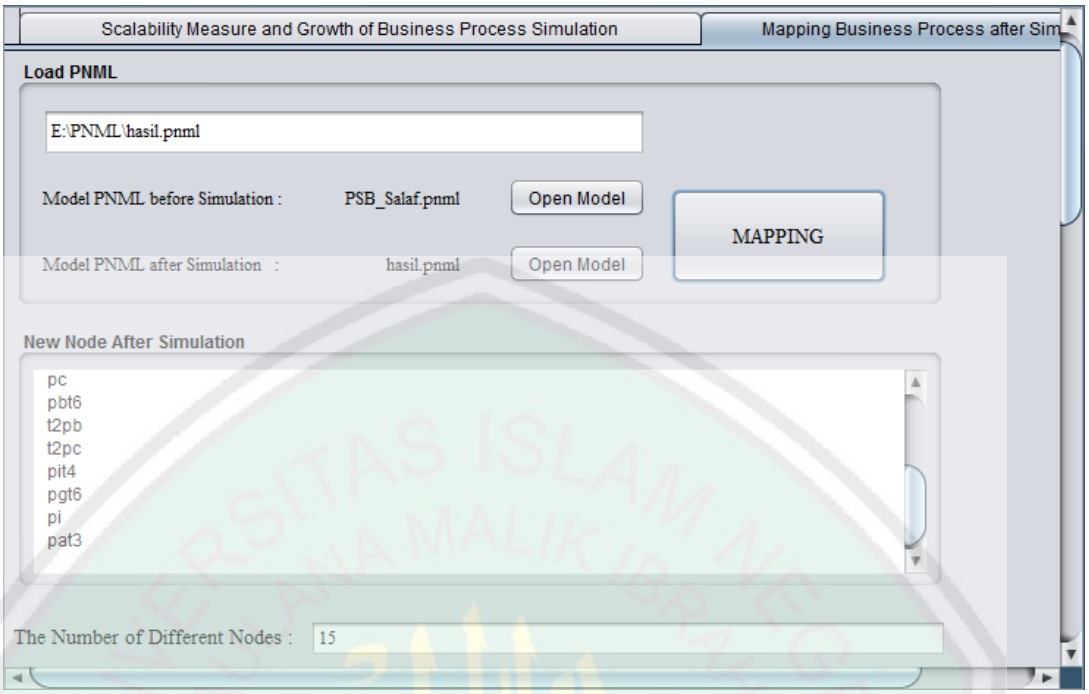
Gambar A 33 *Mapping* model antar PSB *Salaf* dan hasil dari Simulasi model PSB *Salaf*-PSB *Modern*



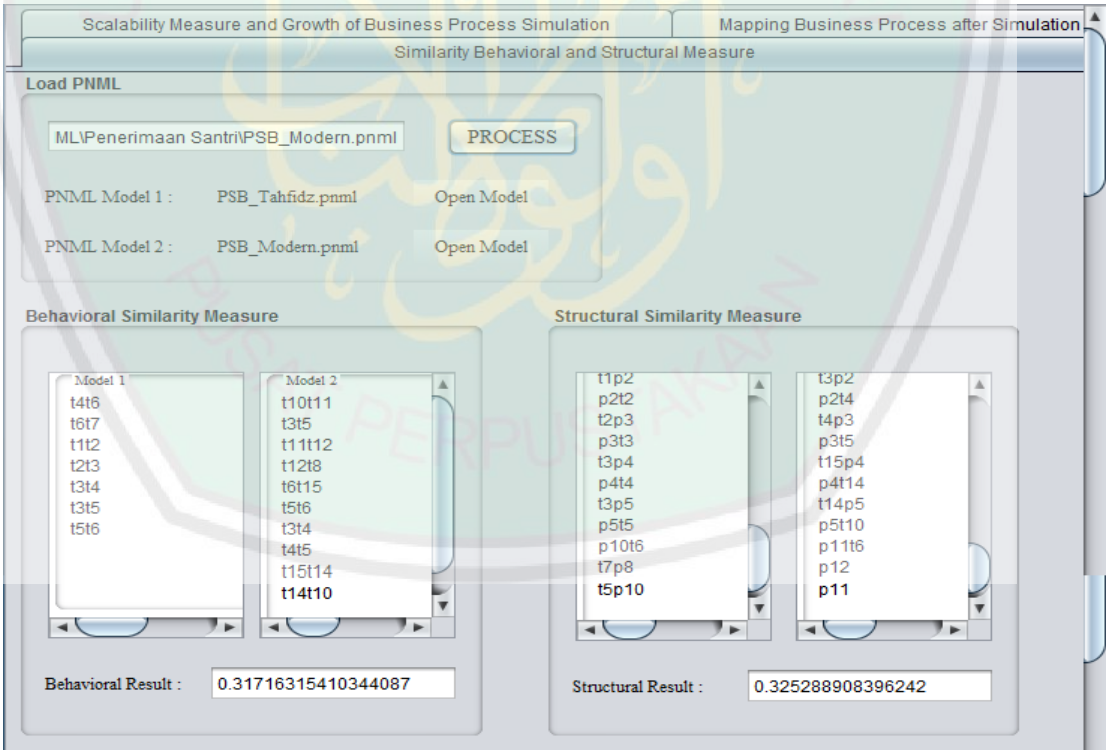
Gambar A 34 Perhitungan *Similarity* antar model PSB *Salaf* dan PSB *Tahfidz*



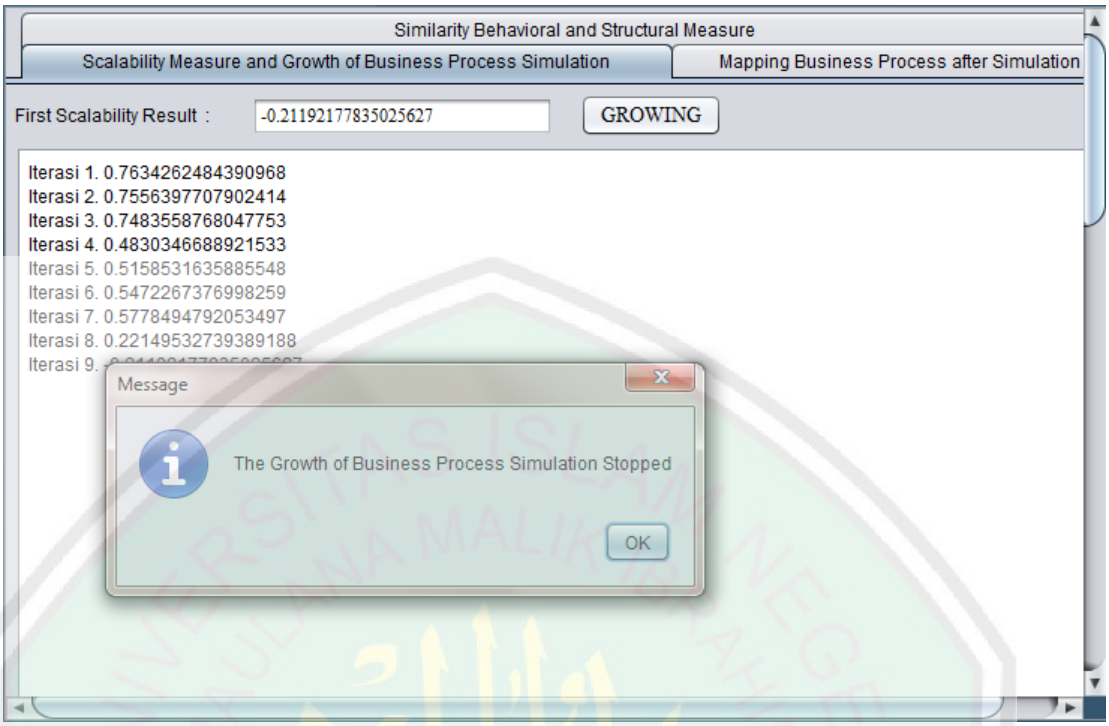
Gambar A 35 Hasil Simulasi Pertumbuhan model antar PSB *Salaf* dan PSB *Tahfidz*



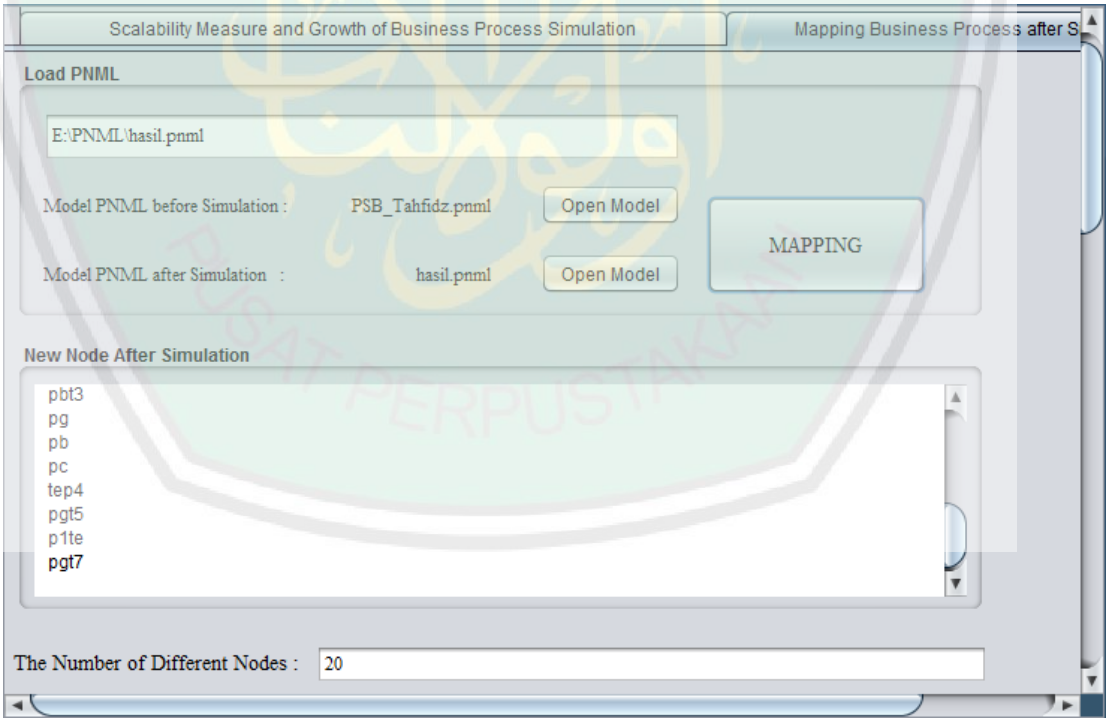
Gambar A 36 *Mapping* model antar PSB *Salaf* dan hasil dari Simulasi model PSB *Salaf*-PSB *Tahfidz*



Gambar A 37 Perhitungan *Similarity* antar model PSB *Tahfidz* dan PSB *Modern*



Gambar A 38 Hasil Simulasi Pertumbuhan model antar PSB *Tahfidz* dan PSB *Modern*



Gambar A 39 *Mapping* model antar PSB *Tahfidz* dan hasil dari Simulasi model PSB *Tahfidz*-PSB *Modern*