Methosisfo : Jurnal Ilmiah Sistem Informasi
ISSN : 2776 – 575X
Vol. 2, No. 2, Oktober 2022, pp. 86-97
http://ojs.fikom-methodist.net/index.php/methosisfo

Survey Paper : Penggunaan Pendekatan Pattern Language pada Pengelolaan Enterprise Architecture (EA)

Sri Agustina Rumapea
Fakultas Ilmu Komputer, Universitas Methodist Indonesia

Info Artikel

Histori Artikel:
Received, 20 Jul, 2022
Revised, 31 Sept, 2022
Accepted, 10 Okt, 2022

Keywords:
Arsitektur enterprise,
Pattern language,
Selected pattern,
Integration pattern

ABSTRAK

Pada penelitian ini akan dibahas mengenai pengelolaan *enterprise architecture* menggunaan pendekatan *pattern language*. Pengelolaan terhadap *enterprise architecture* terkait dengan perubahan/evolusi yang terjadi pada lingkungan (pasar global, perubahan peraturan hokum dan inovasi teknologi) mengharuskan suatu enterprise untuk dapat mengadaptasikan arsitektur enterprisenya. Proses adaptasi dilakukan karena bermanfaat dan diperlukan seringkali diperburuk dengan adanya arsitektur yang rumit. Perubahan terhadap satu artefak lokal seringkali akan memiliki konsekuensi global yang tak terduga dan memiliki potensi yang merugikan. Oleh karena itu mendukung evolusi *enterprise architecture* agar berhasil dengan berfokus pada keselarasan bisnis dan TI merupakan inti dari proses pengelolaan *enterprise architecture*. Pendekatan menggunakan *pattern language* merupakan salah satu cara yang digunakan untuk melakukan pengelolaan terhadap *enterprise architecture*. Pendekatan menggunakan *pattern language* merupakan sebuah pendekatan yang baru di bidang *enterprise architecture*, pendekatan ini menggunakan permasalah berulang yang terjadi dengan solusi yang telah terbukti berhasil menyelesaiakan permasalahan tersebut.

This is an open access article under the CC BY-SA license.

Penulis Koresponden:
Sri Agustina Rumapea,
Fakultas Ilmu Komputer,
Universitas Methodist Indonesia, Medan,
Jl. Hang Tuah No.8, Medan - Sumatera Utara.
Email: sri_rumapea@yahoo.com

I. PENDAHULUAN

Pengelolaan terhadap *enterprise architecture* (EA) telah menjadi tantangan tersendiri bagi sebuah perusahaan. Hal ini disebabkan dengan meningkatnya kompleksitas transaksi bisnis, perubahan regulasi dan meningkatnya ketergantungan perusahaan terhadap teknologi informasi. Bagaimana keselarasan antara bisnis dan Teknologi Informasi (TI) dapat terus terjaga dengan perubahan yang terjadi merupakan tujuan utama dari pengelolaan EA.

Pendekatan berbasis *pattern language* merupakan pendekatan yang dapat dimanfaatkan untuk membuat design terhadap fungsi dari pengelolaan EA. Hal ini karena model yang menggunakan pendekatan ini menyediakan deskripsi yang terstruktur mengenai domain solusi yang dapat digunakan untuk menyelesaikan masalah umum yang berhubungan dengan pengelolaan EA.

Tugas utama dari pengelolaan EA adalah bagaimana mendukung keberhasilan evolusi *enterprise* dengan fokus keselarasan antara bisnis dan pemanfaaran SI dan TI. Pengelolaan terhadap EA juga berusaha untuk mengurangi bahkan menghindari efek samping yang muncul dari proses adaptasi tersebut dan mengambil manfaat dari perubahan yang ada. Terdapat beberapa pendekatan untuk melakukan pengelolaan EA terkait evolusi [1][2], tetapi mereka setidaknya menghadapi salah satu dari permasalah dibawah ini [3]:

1. Pengelolaan terhadap EA sudah diperkenalkan dari awal tidak mempertimbangkan inisiatif yang terjadi di dalam atau luar organisasi.
2. Frame work untuk mengelola EA seperti Zahman , Togaf, dll biasanya terlalu abstrak karena itu seringkali tidak dapat dilaksanakan atau terlalu luas untuk digunakan dalam prakteknya.
3. Kurang memiliki titik awal yang aktual untuk inisiatif pengelolaan EA dimana perusahaan cenderung untuk mengumpulkan persyaratan dari potensi pemangku kepentingan EA dalam organisasi. Mengkonsolidasikan tuntutan dan mengintegrasikan kebutuhan informasi yang dibutuhkan. Pendekataan pengelolaan EA mencakup semua kemungkinan yang muncul, yang akan menuntut sejumlah besar data yang akan dikumpulkan, meskipun sebenarnya hanya sebagian dari itu yang diperlukan untuk mengatasi masalah tertentu.
4. Jika sebuah pendekatan telah digunakan sering kali tidak didokumentasikan sehingga berikutnya tidak diketahui alasan mengapa suatu keputusan tertentu telah diambil.
5. Pendekatan yang diperkenalkan oleh organisasi atau sebuah group standarisasi sebuah pendekatanyang yang tidak incremental dan tidak dapat disesuaikan dengan tingkat *maturity* sebuah enterprise.

Untuk mengatasi masalah tersebut diatas maka diperkenalkan pendekatan dengan menggunakan *pattern* seperti yang sudah di kenal di bidang arsitektur dan *software engineering.* Dalam makalah ini akan dijelaskan pengelolaan EA tekait dengan evolusi yang terjadi dengan menggunakan sebuah pendekatan berbasis *pattern* . Penggunaan *pattern* language dalam EA merupakan hal yang baru . Pendekatan ini digunakan dengan melihat permasalah yang sama dan berulang yang dihadapi oleh enterprise ketika dihadapkan kepada perubahan, Solusi yang terbukti berhasil dalam menghadapi perubahan tersebut selanjutnya akan menjadi bahan pembuatan *pattern* sehingga dapat digunakan kembali sebagai panduan bagi enterprise lain.

II. DEFINISI ENTERPRISE ARCHITECTURE (EA)

Enterprise merupakan sebuah entitas yang komplek dan dinamis, terus bertambah matang dan mengalami evolusi sehingga framework arsitektur dikembangkan untuk dapat merefleksikan tantangan di lingkungan bisnis yang baru. EA berada pada posisi diantara bisnis dan IT dan harus dapat melayani keduanya dan merupakan sebuah mekanisme untuk berkontribusi terhadap *agility, consistency, compliance dan efficiency* [4]. Enterprise architecture telah menjadi disiplin ilmu untuk bisnis dan Rekayasa Perangkat Lunak. Inti dari pendekatan ini adalah model arsitektur yang bertujuan membuat kompleksitas dari dunia nyata dapat dimengerti dan dikelola oleh manusia. Enterprise architecture dapat dipahami sebagai organisasi fundamental dari sebuah lembaga pemerintahan atau corporation baik secara keseluruhan atau bersama-sama dengan mitra, pemasok dan atau pelanggan (extended enterprise) atau sebagian (divisi, departemen, dsb) dan prinsip dalam perancangan maupun evolusi [5]. Sebuah arsitektur EA yang baik adalah yang mengambarkan kondisi perusahaan dan memberikan arti untuk mendukung perubahan. Sebuah arsitektur yang baik membantu perusahaan untuk melakukan inovasi dan perubahan dengan memberikan kestabilan dan keleluasan untuk berubah [6] .

III. *PATTERN* LANGUANGE

Penggunaan *pattern* dan *pattern language* pada EA relative merupakan konsep yang baru meskipun konsep *pattern* dan EA sudah ada 30 tahun terkahir. *Pattern* merupakan upaya untuk mengambarkan solusi untuk masalah atau praktek dalam konteks yang spesifik yang diambil dari *best practices* dan merupakan solusi yang terbukti [7]. Penggunaan *pattern* memberikan pedoman untuk deskripsi solusi dari untuk analisa, perancangan dan arsitektur yang terkait dengan masalah [7]. Dalam arti yang praktis masing-masing menjelaskan masalah yang berulang terjadi pada konteks tertentu dan kemudian menjelaskan solusi inti yang mendasari masalah sedemikian rupa sehingga solusi tersebut dapat digunakan berulang kali. *Pattern language* adalah struktur dari *pattern* yang menjelaskan bagaimana *pattern* itu sendiri dan *pattern -pattern* lain yang lebih kecil. Selain itu juga termasuk aturan yang ada didalamnya yang menjelaskan bagaimana *pattern* dibuat dan disusun dengan *pattern* yang lain [8].

Buckl dkk. [9] menciptakan istilah *dari Enterprise Architecture Management* (EAM) *pattern.* EAM *pattern* sebagai cara untuk membuat struktur domain dari EAM. EAM *pattern* seperti yang diusulkan [9], secara umum memberikan solusi yang dapat digunakan kembali pada masalah umum yang ada pada pengelolaan EA. Dengan demikian EAM *pattern* menjelaskan solusi dari dunia nyata berdasarkan hasil observasi.

Pendekatan berbasis *pattern* pada pengelolaan EA telah dikembangkan untuk mengatasi masalah yang khas dari pengelolaan EA. Beberapa masalah tersebut diantaranya adalah pedoman yang terlalu abstrak, pedoman yang kurang sesuai untuk dapat diterapkan dan pendekatan yang *monolithic*. Pendekatan yang monolithic adalah pendekatan yang mengejar semuanya untuk dapat berhasil [10]. Terdapat bentuk yang berbeda-beda untuk mendokumentasikan *pattern* dan dapat dikatakan bahwa tidak ada bentuk *pattern* yang ideal dan memilih bentuk *pattern* yang ada atau membangun *pattern* yang baru membutuhkan pengalaman.

A. Bentuk dari template *pattern*

Bentuk dari template *pattern* yang digunakan untuk pengelolaan EA adalah seperti yang diperkenalkan oleh Frank Bushman yang telah berhasil digunakan untuk konteks ini [11] [12] [13].

Tabel 1. Template Frank Buschman

Nama Bagian	Deskripsi
Deskripsi	Ringkasan pendek dari isi
Identifier	Identitas unik dari *pattern* untuk menyederhanakan referensi
Versi	Versi untuk melacak perubahan pada *pattern*
Status	Informasi status mengenai *pattern* : Operational, usang, dalam pengembagan
Contoh	Contoh penggunaan
Context	Situasi dimana *pattern* mungkin dapat digunakan
Masalah	Masalah yang diatasi oleh *pattern* dan diskusi yang terkait dengan itu
Solusi	Prinsip solusi fundamental yang mendasari *pattern*
Implementasi	Panduan untuk mengimplementasikan *pattern*
Variant	Deskripsi singkat dari variant atau spesialisasi dari pola
Penggunaan	Contoh dimana *pattern* digunakan
Konsekuensi	Manfaat yang diberikan oleh *pattern*
Kredit	Kredit yang diberikan pada penulis lain, *reviewer* atau orang yang mempertajam *pattern*

Ernst mengembangkan lebih lanjut gagasan ini kepada *pattern language*. *Pattern language* memperkenalkan tiga jenis pola, yaitu *Metodologi Pattern* (*M - Pattern*), *View Point Pattern*

(*V-Pattern*) dan *Information* (*I-Pattern*), yang digunakan untuk mengembangkan fungsi dari pengelolaan arsitektur enterprise [14].

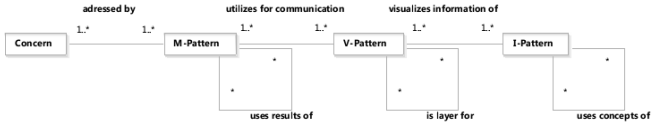
Masing-masing *pattern* memiliki kontribusi yang berbeda terhadap fungsi pengelolaam EA yaitu sebagai berikut:

a) *M-Pattern* menjelaskan metode pengelolaan (dan proses) untuk mem ecahkan masalah spesifik yang terkait dengan EA. Dengan demikian, *M-pattern* memberikan langkah-demi-langkah panduan dan peran informasi secara spesifik tentang apa dan bagaimana melakukannya.

b) V-*Pattern* menjelaskan sudut pandang, yaitu, jenis visualisasi yang akan dibangun dengan *M-Pattern* untuk mengkomunikasikan solusi dan informasi yang relevant tentang EA. *V-pattern* mendokumentasikan solusi yang telah terbukti untuk masalah yang berulang untuk konteks tertentu dalam bentuk *viewpoint* untuk menghasikan sebuah *view*. Dokumentasi *viewpoint* merupakan representasi atau metoda *input* untuk informasi yang dapat disimpan ke dalam satu atau lebih model *information pattern.*

c) *I-Pattern* menjelaskan model konseptual, dimana konsepnya digunakan untuk mendokumentasikan solusi yang merupakan bagian yang relevant dari keseluruhan EA. Model I-*pattern* mendokumentasikan solusi yang pada prakteknya telah terbukti untuk masalah yang berulang pada konteks tertentu dalam bentuk fragment model informasi. Fragment model informasi ini juga mencakup definisi dan deskripsi informasi objects yang digunakan, dokumentasi teknik implementasikan fragmen model informasi dan variasinya dan juga konsekuensi terkait dengan penggunaan fragmen model informasi tersebut.

Dengan menggunakan konseptual UML ketergantungan antara tiga *pattern* tersebut juga dapat digambarkan menggunkan *class diagram* seperti dapat dilihat pada gambar

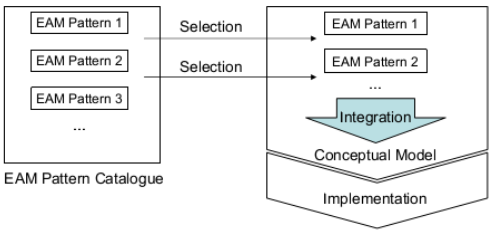


Gambar 1. UML class diagram yang menjelaskan hubungan antara M-*Pattern* , V-*Pattern* dan I-*Pattern* [14]

Kumpulan awal dari *pattern* telah dikumpulkan dari berbagai literature dan praktek, dan juga telah dievaluasi dengan survey yang ekstensif. Survey yang dilakukan melibatkan melibatkan 30 perusahaan menghasilkan EAM Katalog (kumpulan *pattern*) V.1.mengandung 43 *Concern*, 20 M-*Pattern s*, 53- V *Pattern* dan 47 I - *Pattern* [8].

IV. LANGKAH MENGGUNAKAN *PATTERN* UNTUK PENGELOLAAN EA

Penggunaan *pattern* dalam proses pengelolaan EA dimulai dengan melakukan pemilihan terhadap *pattern* yang paling sesuai dengan permasalah yang dihadapi. *Pattern* yang dipilih akan menghasilkan fragment model informasi yang selanjutnya diintegrasikan sebelum pada akhirnya diimplementasikan. Implementasi penggunaan *pattern* dapat dilihat pada gambar dibawah ini.



Gambar 2. Implementasi penggunaan *pattern* pada pengelolaan EA [15]

A. Selection best pattern

Langkah awal untuk menggunakan *pattern* adalah bagaimana memilih *pattern* yang tepat dengan permasalah yang dihadapi berkaitan dengan pegelolaan EA.Terdapat beberapa pendekatan untuk memilih *pattern* terbaik diantaranya adalah:

1. Pemilihan *pattern* berdasarkan pada *concern*

Pemilihan *pattern* untuk pengelolaan EA berdasarkan *concern*. Pendekatan ini yang paling mudah dan banyak digunakan. Cara yang dilakukan untuk memudahkan pemilihan dilakukan dengan mengklasifikasikan *concern*, yaitu :

a) Pemilihan *pattern* dengan menggunakan area topik yang ada pada pengelolaaan EA.
   1. Standarisasi dan keseragaman teknologi
   2. Pengelolaan Business Processes Support Management
   3. Pengelolaan *landscape* aplikasi dengan target perencanaan dan analisa terhadap struktur dan evolusi landscape aplikasi yang berfokus pada *landscape current, planned* dan *target*
   4. Pengelolaan portofolio proyek
   5. Pengelolaan infrastruktur
   6. *Interface, Business Object*, dan *service* manajemen yang berkaitan dengan analisa dan menemukan layanan dalam konteks *service-oriented architectures* (SOA).
   7. Area lain yang tidak tercakup di point sebelumnya.

Pattern katalog yang dikembangkan memetakan concern berdasarkan area topik yang ada di EA.

Tabel 2. Pemetaan concern berdasarkan area topik pengelolaan EA

Technology Homogeneity	C-2, C-4, C-5, C-8, C-9, C-19, C-46, C-50, C-100, C-101
Business Processes	C-54, C-55, C-56
Application Landscape Planning	C-33, C-34, C-35, C-36, C-44, C-86, C-87, C-88, C-89, C-90
Support of Business Processes	C-78, C-80, C-95
Project Portfolio Management	C-29, C-91, C-92
Infrastructure Management	C-41, C-98
Interface, Business Object, and Service Management	C-51, C-52, C-61, C-62, C-64, C-65, C-66, C-67, C-68, C-70, C-71, C-99

Menggunakan area topik yang ada di dalam EA hanyalah salah satu cara untuk dapat mengklasifikasikan *concern* yang ada untuk membantu user dalam memilih *concern* yang tepat. Memilih EAM *pattern* berdasarkan *concern* terdiri dari 4 (empat) proses yaitu memilih:

1. skema kategori,
2. kategori,
3. *concern* yang ada di dalam kategori, dan
4. EAM *pattern.*

b) NASCIO Enterprise Architecture Maturity Model

Variasi yang lain dalam memilih *pattern* berdasarkan *concern* adalah melakukan klasifikasi berdasarkan pada model *maturity*. Contoh yang paling terkenal adalah menggunakan CMMI untuk *development* atau CMMI untuk *services* [13] keduanya dari Software Engineering Institute, Carnegie Mellon University. Pendekatan yang sama juga telah dikembangkan untuk EAs dengan manajemennya NASCIO.

Enterprise Architecture Maturity Model telah dibangun oleh National Association of State Chief Information Officers (NASCIO).

Maturity model tersebut terdiri dari 6 (enam) level yaitu :

EA LEVEL 0 - *No program*
EA LEVEL 1 - *Informal program*
EA LEVEL 2 - *Repeatable program*
EA LEVEL 3 - *Well-defined program*
EA LEVEL 4 - *Managed program*
EA LEVEL 5 - *Continuously improving vital program*

Untuk setiap level *maturity* terdapat 8 (delapan) kategori dimana masing-masing kategori mencakup informasi apa yang diharapkan dari sebuah organisasi.

1. Administrasi: Peran tata kelola dan tanggung jawab
2. Perencanaan: *Road map* dari program EA dan perencanaan implementasi
3. *Framework*: Proses dan *template* yang digunakan untuk EA
4. Cetak biru: Kumpulan dari standar aktual dan spesifikasi
5. Komunikasi: Edukasi dan distribusi EA dan cetak biru secara detil.
6. Kepatuhan: Kepatuhan terhadap standar yang diberikan proses dan element lain dari EA dan proses untuk mendokumentasikan dan melacak variasi lain dari standar.
7. Integration: *touch-poins* dari proses pengelolaan EA
8. Keterlibatan: Dukungan program EA terhadap organisasi.

Empat proses yang harus dilalui untuk memilih *pattern* berdasarkan pada model *maturity* adalah:

1. memilih model *maturity*,
2. menentukan tingkat *maturity* yang akan dicapai,
3. memilih *concern* yang ada di dalam *level maturity* tersebut, dan
4. memilih *pattern* yang ada pada *concern* tersebut, step ini sama dengan pendekatan sebelumnya

c) Extended Enterprise Architecture Maturity Model (E2AMM) V.2.0

Extended Enterprise Architecture Maturity Model (E2AMM) v2.0 [IF04] telah dibangun oleh *Institute for Enterprise Architecture Developments (IFEAD).* Sama dengan model maturity sebelumnya maturity pada EA dan peningkatan procedural dibagi menjadi 6 (enam) level, yaitu:

Level 0: *No extended EA*
Level 1: *Initial*
Level 2: *Under development*
Level 3: *Defined*
Level 4: *Managed*
Level 5: *Optimized*

Dalam masing-masing level tersebut dirinci mengenai hal yang harus dilakukan oleh enterprise di level tersebut menggunakan 11 (sebelas) topik kategori, yaitu.

1. *Business & technology strategy alignment*
2. *Extended enterprise involvement*
3. *Executive-management involvement*
4. *Business units involvement*
5. *Extended EA program office*
6. *Extended EA developments*
7. *Extended EA results*
8. *Strategic governance*
9. *Enterprise program management*
10. *Holistic extended EA*
11. *Enterprise budget & procurement strategy*

*2.*Pemilihan *pattern* berdasarkan *Language Grammars* dan *Design Space Analysis*

Pendekatan ini merupakan cara formal yang dilakukan untuk memilih *pattern*. Zdun [16] menggunakan *Pattern Language Grammar* untuk menemukan *pattern* yang benar. Melakukan elaborasi terhadap deskripsi *pattern* yang mengandung semua informasi yang relevan untuk perancangan keputusan yang informal. Informasi tersebut adalah:

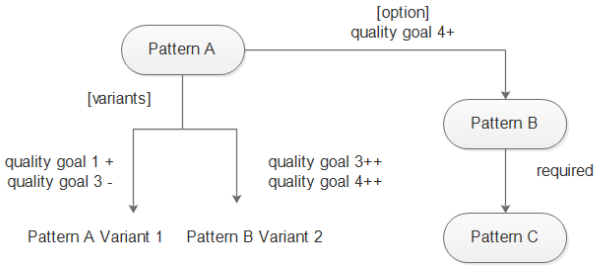
- *Functional Requirement:* Setiap *pattern* menjelaskan kebutuhan fungsional yang dipenuhi dan menjelaskan solusi praktis yang telah berhasil dilakukan mengatasi masalah dalam konteks tersebut.
- *Quality goals:* Setiap *pattern* mengandung 2 (dua) bagian yaitu *Forces* (kekuatan) dan *Consequence* (konsekuensi) yang menjelaskan pengaruh potensial terhadap kualitas tujuan
- *Pattern* variants: Variasi *pattern* yang dapat digunakan sebagai solusi.
- Related *pattern*: Hubungan dengan *pattern* lain, dimana *pattern* lain yang perlu dipertimbangkan digunakan ketika menggunakan sebuah *pattern* .

Menurut Zdun informasi ini saja tidak cukup untuk merancang sebuah keputusan yang sistematis oleh karena itu untuk mengatasi masalah tersebut Zdun melakukan formalisasi terhadap *pattern relationship* dalam sebuah *grammar* dan memberikan catatan pada grammar dengan pengaruh yang diberikan pada kualitas tujuan.

Untuk menunjukan pengaruh terhadap kualitas Zdung megusulkan notasi seperti dibawah ini :

++	:	Diperkirakan pengaruh yang sangat positif terhadap terhadap kualitas tujuan
+	:	Diperkirakan pengaruh positif terhadap kualitas tujuan
0	:	Diperkirakan tidak terdapat pengaruh terhadap kualitas tujuan
-	:	Diperkirakan pengaruh negatif terhadap kualitas tujuan
--	:	Diperkirakan pengaruh yang sangat negative terhadap kualitas tujuan

Contoh penggunaan notasi untuk menunjukan pengaruh *pattern* terhadap kualitas tujuan dapat dilihat pada gambar dibawah ini.



Gambar 3. Contoh dari *Sequence pattern diagram*

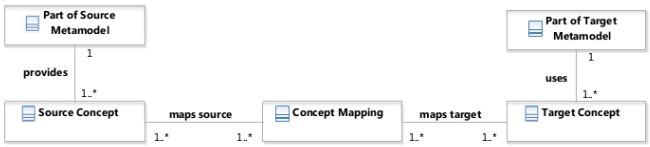
Hasilnya adalah urutan *pattern* yang secara drastis mengurangi kombinasi *pattern* yang mungkin. *Pattern sequence diagram* merupakan aktivitas yang dilakukan untuk melakukan *mapping pattern* pada sebuah *design space*.

B. Integration

Proses integrasi terhadap *pattern* yang telah dipilih merupakan aspek penting dalam pengelolaan .Sebelum akhirnya diimplementasikan fragment *I-pattern* diintegrasikan terlebih dahulu. Ada beberapa pendekatan untuk melakukan integrasi yang dilakukan di bidang Software Engineering dan basis data terdistribusi. Kuhn menjelaskan pendekatan yang mirip dengan yang dijelaskan oleh perbedaannya adalah Ku lebih detail dan membuat sebuah system *pattern* [17]. Pendekatan ini yang digunakan untuk melakukan integrasi terhadap fragement *I-pattern*.

a. Integrasi dengan *Concept mapping*

Cara yang paling intuitif untuk mengintegrasikan 2(dua) fragment *pattern* . S.Buckl mengusulkan untuk mengidentifikasikan 1(satu) atau lebih *class* yang identik dari kedua *pattern* . Class ini kemudian dapat digunakan sebagai titik integrasi [12]. Kuhn mengambarkan pendekatan yang sama yang disebut sebagai Concept – Mapping – *Pattern* seperti yang ditunjukan dengan gambar dibawah ini [17].

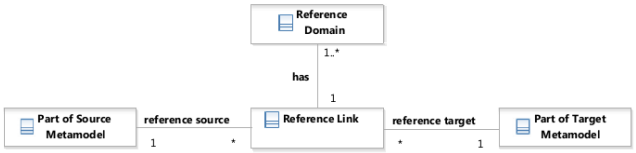


Gambar 4. Conceptual Class Diagram untuk integrasi dengan *concept mapping* [17]

Sebuah *Concept Mapping* dapat didefinisikan untuk mengintegrasikan satu atau lebih *source concept* dari 1(satu) model informasi dengan 1(satu) atau lebih *concept target* dari model yang lain.

b. Integrasi dengan Reference

Cara kedua untuk mengintegrasikan *pattern* adalah dengan memanfaatkan referensi. Di dalam referensi diperkenalkan antara 1(satu) konsep dari *source information model* dan 1(satu) konsep dari *target information model* [17].

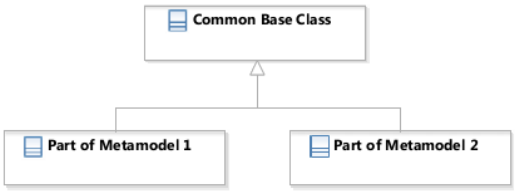


Gambar 5. *Conceptual Class diagram* untuk integrasi dengan referensi [17]

Conceptual class diagram dari Kuhn juga menambahkan *Referensi class* yang disebut dengan Referensi domain yang digunakan untuk mendefinisikan kardinalitas dari element sumber ke element target.

c. Integrasi dengan *Common Base Class*

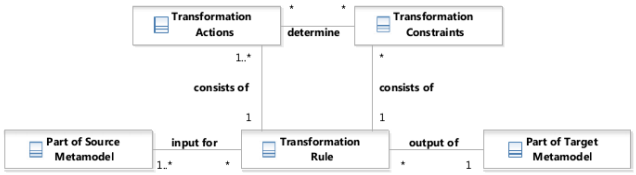
Sebuah pendekatan yang sama untuk mengintegrasikan I-*pattern* dengan menggunakan referensi adalah untuk menentukan common base class yang dapat digunakan untuk mengintegrasikan elemen. Gambar dibawah ini menunjukkan *Common base class* didefinisikan untuk bagian metamodel 1 dan metamodel 2.



Gambar 6. *Conceptual Class Diagram* dengan *Common base class* [21]

d. Integrasi dengan Transformasi

Dalam kasus dimana terdapat 1(satu) atau lebih metamodel yang digunakan secara independent integrasi dengan transformasi ini dapat digunakan. Transformasi digunakan untuk membuat metamodel baru dengan menggunakan aturan transformasi yang telah didefinisikan [21]



Gambar 7. Conceptual class diagram untuk integrasi dengan transformasi [17]

V. CONTOH IMPLEMENTASI *PATTERN* LANGUAGE

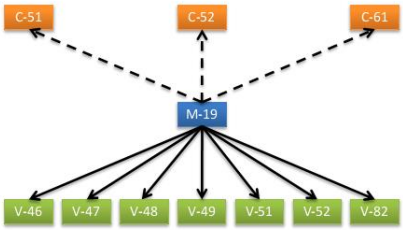
Langkah pertama adalah melakukan pemilihan *pattern* yang sesuai, pada makalah ini pemilihan *pattern* terbaik menggunakan pendekatan berdasarkan concern yang berdasarkan pada area topik yang ada di EA. Area topik yang dipilih adalah *Interface, Business Object* dan *Service Management dengan Concern*. C.61. Objek bisnis yang mana dipertukarkan diatara *interface*. Untuk *concern* tersebut diatas menggunakan metodologi M.19 yaitu *Management of Business Object* dimana M.19 *Pattern* mencakup manajemen dari objek bisnis, atribut dan relasinya.

Tabel 3. M-19. Manajemen Bisnis Objek [11]

Id	**M-19**
Name	Management objek bisnis
Ringkasan	M-*Pattern* mencakup manjemen dari objek bisnis, atribut dan relasinya

Dari M.19 diturunkan menjadi V-*pattern* , dimana yang terlibat adalah:

1. V - 46 *Business Object ER diagram*
2. V - 47 *Business Object class diagram*
3. V - 48 *Cluster map visualization business object flow between business application*
4. V – 49 *Communication table*
5. V – 51 *Process Overview*
6. V – 52 *Business level Communication Overview*
7. V – 82 *Business Object Flows*



Gambar 8. Hubungan antara *Concern* , M-*Pattern* dan V-*Pattern untuk C.61* [11]

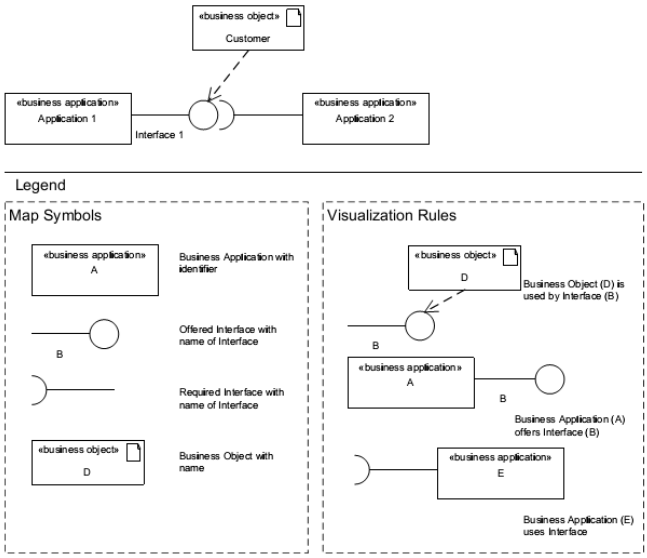
Tujuan dari manajemen bisnis objek *manajement business object* adalah untuk menjelaskan lebih rinci *pertukaran business object* antara *business process, business services* atau *business aplications*. V-*pattern* (V-46 dan V-47) menggunakan notasi yang berbeda tetapi dapat saling dipertukarkan karena mereka memvisualisasikan hal yang sama *yaitu business objects, attributes* dan *relationships*. V-*pattern* (V-48, V-49, V-51, V-52 dan V-82) juga dipertukarkan karena mereka dapat digunakan untuk menunjukkan ketergantungan antara aplikasi bisnis atau layanan bisnis. Dari masing-masing V-*pattern* akan diturunkan menjadi I – *pattern*, yang selanjutnya I –*pattern* tersebut diintegrasikan. Dalam makalah ini hanya akan dituliskan 1 (satu) V*pattern* yaitu untuk V-82.

Tabel 4. V.82. Aliran Objek Bisnis [11]

Nama	**Aliran Object Bisnis**

Id	V.82
Alias	
Ringkasan	Visualisasi V-*pattern* menjelaskan bagaimana objek bisnis dipertukarkan diantara aplikasi bisnis

Solusi :



Gambar 9. Viewpoint V-82 Aliran Objek Bisnis [11]

V-*pattern* ini menunjukan bagaimana objek bisnis dipertukarkan diantara aplikasi bisnis. V-82 ini berdasarkan I-*pattern* I-63 dan I-82.

Tabel 5. I-82 [11]

Id	**I-82**
Alias	
Ringkasan	

Solusi:

Gambar 10. Fragment Model Informasi I-82 [11]

Business Aplication: sebuah bisnis aplikasi merupakan sebuah system perangkat lunak yang merupakan bagian darisistem informasi dari sebuah organisasi.

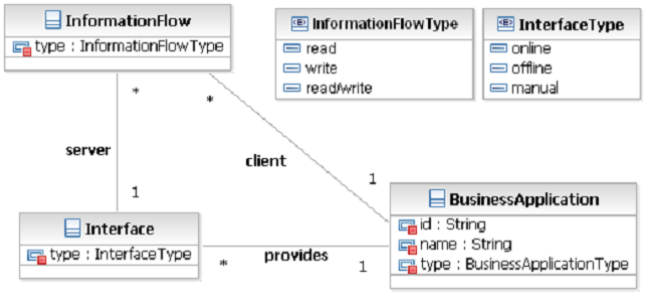
Business Object: Sebuah objek bisnis merepresentasikan entitasi bisnis yang digunakan pada selama eksekusi proses bisnis yang melakukan operasi (CRUD).
Interface : Antar muka aplikasi bisnis.

Tabel. 6. I-63 [11]

Id	**I-63**
Name	Antar muka dan aliran informasi
Alias	

Ringkasan

Solusi



Gambar 11. Fragment Model Informasi I-63 [11]

VI. KESIMPULAN

Pendekatan berbasis *pattern* merupakan pendekatan yang dapat dimanfaatkan untuk membuat *design* terhadap fungsi dari pengelolaan EA. Ini karena model yang menggunakan pendekatan ini menyediakan deskripsi yang terstruktur mengenai domain solusi yang dapat digunakan untuk menyelesaikan masalah umum yang berhubungan dengan pengelolaan EA. Dari hasil survey paper yang dilakukan maka berikutnya akan dilakukan formalisasi terhadap pattern language yang ada untuk salah satu frame work EA.

REFERENCES

[1] C. Braun and R. Winter, "A Comprehensive Enterprise Architecture Metamodel and Its Implementation Using a Metamodeling Platform 2 Modeling Enterprise Architecture."

[2] S. Buckl, F. Matthes, and C. M. Schweda, "From EA management patterns towards a prescriptive theory for designing enterprise-specific EA management functions Outline of a research stream," no. 1987, pp. 67–78, 2010.

[3] S. Buckl, F. Matthes, I. Monahov, S. Roth, C. Schulz, and C. M. Schweda, "Enterprise Architecture Management Patterns for Company-wide Access Views on Business Objects Modern application landscapes consist of a multitude of inter-connected business applications exchanging data in many ways . These business applications are used," vol. 2, no. 3, pp. 1–12, 2012.

[4] J. Schelp and R. Winter, "Language Communities in Enterprise Architecture Research," 2009.

[5] B. R. Winter and R. Fischer, "Article Essential Layers , Artifacts , and Dependencies of Enterprise Architecture," no. May, pp. 1–12, 2007.

[6] R. Foorthuis, M. van Steenbergen, S. Brinkkemper, and W. A. G. Bruls, "A theory building study of enterprise architecture practices and benefits," *Inf. Syst. Front.*, vol. 18, no. 3, pp. 541–564, 2016.

[7] S. Patterns, *Software Patterns*. .

[8] S. Buckl, A. M. Ernst, J. Lankes, F. Matthes, and C. M. Schweda, "Enterprise Architecture Management Patterns Exemplifying the Approach," vol. 93, 2008.

[9] S. Buckl, A. M. Ernst, J. Lankes, K. Schneider, and C. M. Schweda, "A Pattern based Approach for constructing Enterprise Architecture Management Information Models."

[10] S. Buckl and C. M. Schweda, "On the State-of-the-Art in Enterprise Architecture Management Literature," 2011.

[11] D. Ori, "A rule-based approach to business-IT misalignment symptom detection," *Int. Conf. Software, Knowl. Information, Ind. Manag. Appl. Ski.*, vol. 2017-Decem, no. December 2017, 2018.

[12] F. Matthes and C. Neubert, "Wiki4EAM – Using Hybrid Wikis for Enterprise Architecture Management," p. 4503, 2011.

[13] A. Lau *et al.*, "EA Management Patterns for Smart Networks."

[14] E. Denert-stiftungslehrstuhl, "Enterprise Architecture Management Pattern Catalog," no. February, 2008.

[15] M. Taleb and O. Cherkaoui, "Pattern-Oriented Approach for Enterprise Architecture : TOGAF Framework," vol. 2012, no. January, pp. 45–50, 2012.

[16] U. Zdun, “Systematic Pattern Selection Using Pattern Language Grammars and Design Space Analysis.”
[17] H. Kühn, F. Bayer, and D. Karagiannis, “Enterprise Model Integration,” 2003.