Optimizing Enterprise Architectures Using Linear Integer Programming Techniques

Simon Hacks and Horst Lichter[1]

Abstract: Within this paper, we present a technique to optimize the relations between two adjacent layers of Enterprise Architectures (EA). Therefore, we suggest to interpret the constraints between these two layers as triangles, where a needed capability of an upper layer element is realized by a lower layer element. This eases the communication of the optimization model e.g. to the management. Moreover, we propose a mapping between the elements of our technique to the widely accepted ArchiMate notation to enable the application of our technique in existing organizations.

Keywords: Enterprise Architecture, Enterprise Architecture Management, Linear Integer Programming, Optimization, ArchiMate

1 Introduction

According to ISO/IEC/IEEE 42010:2011 [III11] architecture is defined as the "fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution". Consequently, Enterprise Architecture (EA) is about organization's elements and their relations.

To differentiate between the architecture model and its management we stick to the definitions given by [PS04] and [Ro94] defining EA as a model of an organization describing its primary components and the relations and interactions among them.

Enterprise Architecture Management (EAM) is a discipline comprising all functions which are related to the EA, like maintaining the EA itself, but also providing information gathered from the EA. EAM has developed to an established discipline in industry and it is also a current topic in research.

The origins of EAM date back to 1987 when Zachman published the "framework for information systems architecture" [Za87]. Since then, it was recognized that a limited view, focusing on the information system architecture only, is not sufficient. Therefore, present EAM frameworks include the management of business capabilities, infrastructure components, and information structures, too [WF06].

[1] RWTH Aachen University, Research Group Software Construction, Ahornstraße 55, 52074 Aachen, {hacks, lichter}@swc.rwth-aachen.de

doi:10.18420/in2017_59

Obviously, the beforehand stated properties of EA allow to interpret the EA as a graph, where EA's elements are vertices and their relationships are edges. The different layers of EA can be represented by sets of vertices.

As we can represent an EA as a graph, we can apply existing graph algorithms to solve problems within the domain of EA. For instance, the well known Travelling Salesperson Problem (TSP) can be formulated as a Linear Integer Program (LIP) on a graph [Ap11, pp. 1-5]. This universal formulation allows to apply this solution to other domains like genome sequencing, drilling problems, or data clustering [Ap11, pp. 59-70].

Previous research shows that information technology (IT) has become more important for business during the last decades and changes business models dramatacly [BL08, OS00]. Simultaneous, the IT pervades the business more and more and becomes ubiquitous [VSM10]. This rises also the complexity of the information systems and their interrelations [LW01]. With rising complexity of the information systems it becomes harder to ensure the IT/business alignment.

One way to achieve the IT/business alignment is EAM [AW09, Pl07]. Corresponding to the maturity of the EAM [AWW12] different goals are aimed. For example, the IT landscape should be consolidated to reduce costs or to identify functional redundancies between business and IT. To support those goals different techniques are feasible.

In the following, we propose an graph-based technique to improve EAs with respect to loose coupling, minimal amount of elements, and minimal operation costs, using LIP. We use the metaphor of triangles to build up our LIP, which is more intuitively compared to other techniques (e.g. [Fr10, Gi12]). Those techniques optimize EAs using graphs and LIP.

The rest of the paper is structured as follow: In Section 2 we discuss related work and demonstrate the improvement potentials we overcome with our technique presented in Section 3. Afterwards, we propose a mapping between the elements used in our technique and ArchiMate to ease the application in organizations, before we come to the conclusion and point out further extensions to our technique.

2 Related Work

Beforehand, we discussed the need for techniques to support EAM. Within the scientific community, different techniques were proposed. In the following, we will present those techniques using mathematical programming in EAM related topics:

[Gi12] present a multiobjective program to solve the "information system architecture evolution management problem"which handles the issue of scheduling the replacement of existing services with new services without discontinuity. Therefore, they model departments, existing services, new services, and IT modules. Moreover, they link services and modules, departments and services and attach certain costs to changes on the model. Each department obtains budgets to finance new modules and retire old ones. Using this model, they maximize the business gain and the killing gain with respect to different constraints.

The approach of [ST03] is slightly different compared to [Gi12]. Where [Gi12] assume a single organization, [ST03] optimize the IT landscape in a supply chain environment. Furthermore, they do not assume killing costs but various purchasing and integration costs at varying points in time and optimize via a LIP the net profit.

[Fr10] want to find the optimal allocation of IT systems to processes with respect to needed functionalities. These functionalities are needed by processes and fulfilled by certain systems. In a second step the functionalities are dissolved and the processes and systems are connected directly. These connections describe the as-is state. Taking change costs and operations costs into account, they optimize the allocation of the IT systems via a binary integer program.

All these approaches have in common that they focus on the mathematical formulation and solving the model. Thereby, they neglect the aspect of communication to EAM's stakeholder like managers. These managers may not have the ability to understand intuitively the mathematical models. Therefore, there is a need to offer a more intuitive and understandable approach to model EA in order to identify deficiencies that can be improved. Furthermore, such an approach should increase its acceptance and applicability in industry.

3 An Optimization Model for Enterprise Architectures

3.1 Foundations

As presented by [WF06], an EA is structured in a layered manner. Each layer contains different architectural elements, which have relations to other elements of the same layer as well as relations to architectural elements of adjacent layers. We assume that each layer offers capabilities to the layer that is defined on top of it to realize its needed functions and behavior. In the following, we will refer to the layer offering capabilities as the "lower layeränd call the layer using these capabilities the "upper layer". Hereafter, we will describe these layers and elements more vividly:

We want to optimize the relations between the elements of two adjacent layers. For instance, taking an ArchiMate notated EA in account, we may want to optimize the relations between the business layer and the application layer. The business layer contains some business functions, which should be realized by several application components. Therefore, each business function is connected to all necessary capabilities and each application component is connected to all capabilities it realizes. Based on the relations between the architectural elements and the capabilities, we suggest a technique to find the "optimal"relations between the elements of the adjacent layers. In our case, "optimal"can stand for e.g. a minimal amount of elements or minimal costs.

To be more concrete, assume that the business layer contains the business function *Personnel Management*, which requires e.g. the capabilities *Hire Employee*, *Calculate Wage*, and *Record Written Warning*. These capabilities can be realized either by two different application components or one single application component. Depending on the understanding of “optimal", different solution scenarios are feasible.

Following, we will use a simplified EA model sketched in Figure 1. It contains two adjacent layers and a set of capabilities. Each architectural element of the upper layer, ul_i, is related to several capabilities, c_j, which are needed to realize ul_i.

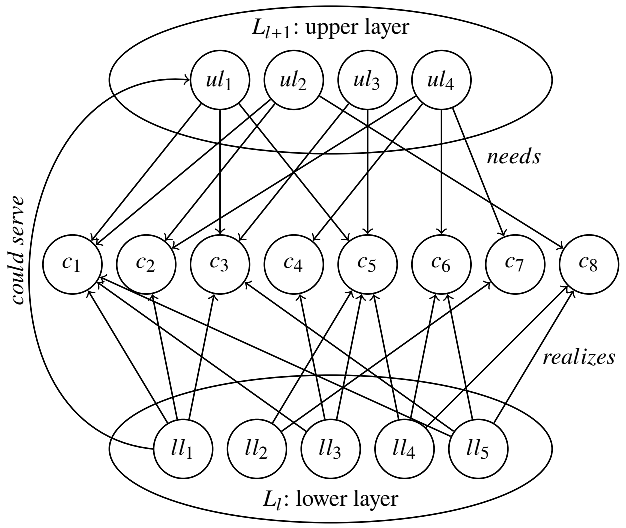


Fig. 1: Example Enterprise Architecture.

Having a look from the lower layer, there are several architectural elements, ll_k, which realize different capabilities, c_j, defined by the relation ll_k to c_j. Based on this structures, we want to optimize the relations between the upper and the lower layer elements to different subjects.

Before we dive into the optimization, we define the used EA model more formal:

An enterprise architecture is a quadruple $EA = \mathcal{L}, \mathcal{C}, E, R$ comprising an ordered set $\mathcal{L}$ of layers, a set $\mathcal{C}$ of capability sets, a set E of architectural elements, and a set R of relations.

Each layer $L \in \mathcal{L}$ consists of architectural elements and layers are disjunct:

$$L_i \cap L_j = \emptyset \quad | \quad \forall L_i, L_j \in \mathcal{L}, i \neq j \tag{1}$$

For each two adjacent layers, L_l and L_{l+1}, there is a set of capabilities $C_{L_l,L_{l+1}} \in \mathcal{C}$. Each capability describes some specific behaviour, which can be only offered by a certain layer; therefore, all these capability sets are disjunct:

$$C_i \cap C_j = \emptyset \quad | \quad \forall C_i, C_j \in \mathcal{C}, i \neq j \tag{2}$$

A relation is a tuple of an architectural element and a capability:

$$r \in R \subseteq \{e, c : e \in L_l, L_{l+1}, c \in C_{L_l, L_{l+1}}\} \tag{3}$$

We want to find the “optimalßet of relations between the elements of the upper and the lower layer using beforehand stated foundations. Therefore, we create a complete bipartite graph between the layers L_l and L_{l+1}:

$$r_I^{l,l+1} \in R_I^{l,l+1} \equiv \{ul, ll : ul \in L_{l+1}, ll \in L_l\} \,|\, L \in \mathcal{L} \tag{4}$$

This intermediately introduced relations represent all possible connections between the two adjacent layers with no respect to constraints given by the relations between architectural elements and capabilities.

3.2 Applying the Modeling Approach to ArchiMate

To illustrate our EA modelling approach based on layers and capabilities, we applied our approach for ArchiMate modelled EAs, as ArchiMate [La04] is a widely accepted and used EA modelling language.

For the sake of simplicity, we considered only the element types of three ArchiMate layer: business layer, application layer, and technology layer. Those layers contain element types like processes or software, which are related to each other. Within the chosen layers, we disregarded the passive structure element types, since they all represent some kind of information objects in different ways, which cannot be linked to some kind of capabilities.

Table 1 shows the mapping of ArchiMate modelling element types to the three considered layers and the respective capabilities.

We mapped the Application Function element type of ArchiMate to a capability, because it can describe a requirement the business has for an application. Business Process and Business Function, modeling the overall structure of an organisation and how the organisation realizes the value chain, can be obviously mapped to element types of the upper layer. These element types may need some application support from the Application Layer, which can be represented by the element types Application Component, Application Collaboration, or Application Process, realizing the needed capabilities, i.e. Application Functions.

Similarly to Application Function, we mapped the Technology Function element type to a capability describing the needs of the element types of the Application Layer. Furthermore,

	ArchiMate Element Type											
	Business Process	Business Function	Application Component	Application Collaboration	Application Function	Application Process	Node	Device	System Software	Technology Collaboration	Technology Function	Technology Process
Business Layer	●	●										
Capability					●							
Application Layer			●	●		●						
Capability											●	
Technology Layer							●	●	●	●		●

Tab. 1: Suggested Mapping for ArchiMate Element Types to Proposed Sets.

Technology Collaboration and Technology Process can be mapped to element types of the lower layer. Unfortunately, there is no element type called “Technology Component". Instead, the element type Node can be used to offer certain functionalities to the Application Layer. Furthermore, it aggregates Device, which represents hardware resources, and System Software, which represents software resources. Moreover, these aggregated element types are also needed to realize the functionality of the Application Layer and, consequently, are mapped to elements of the lower layer.

3.3 Different Optimization Subjects

Beforehand, we formulated a formal description of an EA model. Based on this formal description, we postulated a mapping for ArchiMate. Next, we will define several different objectives to optimize the EA model.

Optimizing with Respect to Minimal Coupling

Loose coupling between two components eases the interchangeability [SM96]. Hence, the exchange of a single component will not be as expensive as in a highly coupled system. This holds also for EAs. Consequently, managers might be interested to find the minimal coupling between EAs elements.

Optimizing the EA with respect to a minimal coupling between two adjacent L_l and L_{l+1}, we have to minimize the amount of used intermediate relations, $r_I^{l,l+1}$. Consequently, those $r_I^{l,l+1}$ represent the optimization variables of our LIP:

$$min \quad x_{r_I^{l,l+1}}, \tag{5}$$

where $x_{r_I^{l,l+1}} \in \{0,1\}$ and $x_{r_I^{l,l+1}} = 1$ means that $r_I^{l,l+1}$ is in the optimal solution and $x_{r_I^{l,l+1}} = 0$ means that it is not in the optimal solution.

To guarantee the restrictions sketched in Figure 1 we have to define several constraints. To ensure for each upper element that every capability which is needed will be served in the solution, there has to be at least one relation between an upper layer element and a lower layer element which supports this capability:

$$\underset{r_I \in S^R_{ul_i,c_j}}{} x_{r_I^{l,l+1}} \geq 1 \quad | \forall S^R_{ul_i,c_j} \in \mathcal{S}^R, \tag{6}$$

with $\mathcal{S}^R \ni S^R_{ul_i,c_j} \subseteq R_I^{l,l+1}$. Where $\mathcal{S}^R$ contains all intermediate relations which are taken into account for the optimization and $S^R_{ul_i,c_j}$ contains all intermediate relations between ul_i and ll_k, where ll_k has a relation to c_j:

$$S^R_{ul_i,c_j} = \{ul_i, ll_k \in R_I^{l,l+1}\} \quad | \forall ul_i, c_j \in R, \forall ll_k, c_j \in R \tag{7}$$

To easily create all $S^R_{ul_i,c_j}$ we can create a maximum flow problem [HR55] for each connection between the upper layer elements and the related capabilities. Therefore, we introduce additional nodes: a source s, and a sink t. We connect s to ul_i and replace all upper layer elements in intermediate relations, $r_I^{l,l+1}$, by t. Furthermore, we introduce the capacity of a relation as a mapping $\mathfrak{c} : R \cup R_I^{l,l+1} \cup s, ul_i \to \mathbb{R}^+$ denoted by $\mathfrak{c}u, v$. The capacity defines the maximum amount of flow which can pass a relation.

A flow of a relation is a mapping $\mathfrak{f} : R \cup R_v \cup s, ul_i \to \mathbb{R}^+$ denoted by $\mathfrak{f}u, v$ with subject to two constraints. First, the flow cannot exceed its capacity:

$$\mathfrak{f}u, v \leq \mathfrak{c}u, v \quad | \forall u, v \in R \cup R_v \cup s, ul_i \tag{8}$$

Second, the sum of the entering flows must equal the sum of the leaving flows:

$$\underset{u:u,v \in R}{} \mathfrak{f}u, v = \underset{u:u,v \in R}{} \mathfrak{f}v, u \,|\, \forall v \in L_l \cup L_{l+1} \cup C_m \tag{9}$$

Before we apply e.g. the algorithm of [FF56] or [Di70] to solve the maximum flow problem, we set the capacity of all intermediate relations, $r_I^{l,l+1}$, to 1 and the capacity of all other relations to infinity. After the application, we add all intermediate relations whose flow is 1 to $S^R_{ul_i,c_j}$.

Figure 2 shows a subset of the EA in Figure 1. It contains ul_1 and all related capabilities, all ll_k which offer those capabilities, and the intermediate relations represented by the dashed lines. For instance, solving the maximum flow problem and creating the necessary set for the relation between ul_1 and c_1 leads to

$$S^R_{ul_1,c_1} = \{ul_1, ll_1, ul_1, ll_3\}, \tag{10}$$

which, consequently, creates the following constraint in our LIP:

$$x_{ul_1,ll_1} + x_{ul_1,ll_3} \geqslant 1. \tag{11}$$

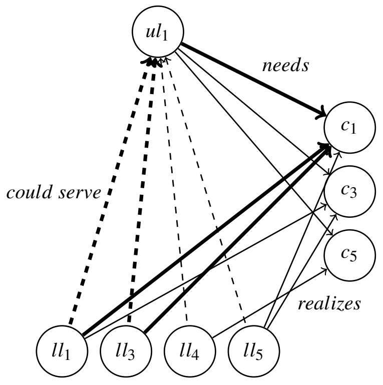


Fig. 2: Visualization of $S^R_{ul_1,c_1}$.

Those constraints ensure that in the final solution, proposed by the LIP solver, at least one intermediate relation between the upper and the lower layer element is used, which are both related to the same capability. This leads to a distinctive structure stressed out in Figure 2 by the thicker lines: a triangle. These triangles describe the constraints which have to be included in a later solution.

Optimizing the Amount of Needed Lower Layer Elements

The beforehand stated foundations can also be used to optimize the amount of needed lower layer architectural elements. Managers might want to have the minimal amount of elements in a certain layer to reduce the needed knowledge to maintain those elements.

To achieve the minimal amount of elements, we have to adjust the optimization function (5) and the constraints (6) from relations to lower layer elements. This leads to following optimization function:

$$min \quad \underset{ll\in L_l}{} x_{ll}. \tag{12}$$

The constraints are constructed by using the lower layer elements as well:

$$\underset{ll \in S^E_{ul_i,c_j}}{} x_{ll} \geq 1 \quad | \forall S^L_{ul_i,c_j} \in \mathcal{S}^L, \tag{13}$$

with $\mathcal{S}^E \ni S^E_{ul_i,c_j} \subseteq L_l$ containing all lower layer elements which are part of the intermediate relations between ul_i and ll_k, where ll_k has a relation to c_j:

$$S^E_{ul_i,c_j} = \{ll_k : ul_i, ll_k \in R_v\} \quad | \forall ul_i, c_j \in R, \forall ll_k, c_j \in R \tag{14}$$

According to the aforementioned example, solving the maximum flow problem creates

$$S^E_{ul_1,c_6} = \{ll_4, ll_5\}, \tag{15}$$

which leads to following constraint:

$$x_{ll_4} + x_{ll_5} \geqslant 1. \tag{16}$$

Optimizing Operational Costs

Managers are typically assessed by the costs which occur in their area of response. Consequently, they are often interested in reducing cost without losing functionality.

To optimize the operational costs, we have to apply slightly changes to (12) by introducing an operational cost function $\mathfrak{o} : E \to \mathbb{R}^+$ denoted by $\mathfrak{o}e$:

$$min \underset{ll \in L_l}{} \mathfrak{o} x_{ll}. \tag{17}$$

For simplicity reasons we assume that the operational costs are constant and not affected neither by lower layer elements attached to the considered element nor by relations between elements within the same layer.

3.4 Applying the Optimization Model

Following, we examine our model to check two aspects. First, we check if the proposed solutions are optimal. Second, we like to ensure that our approach is solvable for realistic problems in adequate time.

Exemplary Application

To test our previous formulated LIPs we translated the EA in Figure 1 to Java code and transformed it into a LIP which was solved by LPsolve[2]. Following, we will present the results of the optimization. For reasons of simplicity, we split up the graph in four subgraphs. Each subgraph represents the proposed solution for one upper layer element including the assigned capabilities and lower layer elements.

The results regarding the minimal coupling optimization are presented in Figure 3. The result set contains ten relations between the upper and the lower layer elements, which create the enforced twelve triangles. Furthermore, all five lower layer elements are used.

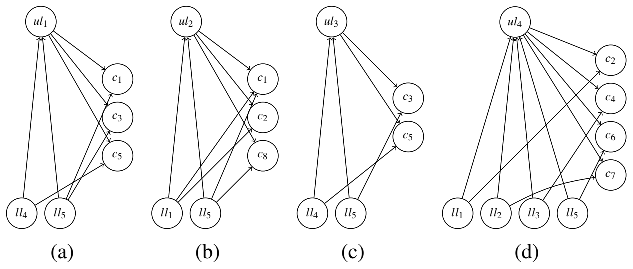


Fig. 3: Solution With Respect to a Minimal Coupling.

Figure 4 sketches the results of of the optimization regarding the minimal amount of lower layer elements. An optimal solution contains four elements as LPsolve suggests to exclude ll_5. Since the LIP optimizes only the amount of lower layer elements, it tells nothing about the concrete assignment of upper to lower layer elements. Therefore, we suggest relations using solid lines as well as all other possible relations using dashed lines.

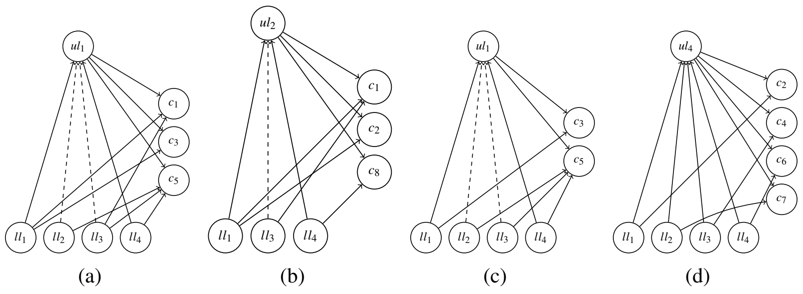


Fig. 4: Solution With Respect to Minimal Lower Layer Element Amount.

To process the optimization with respect to minimal lower layer element costs, we assigned costs stated in Table 2 to the lower layer elements. In Figure 5 we visualized the results. Compared to Figure 4, LPsolve suggests to exclude ll_4 instead of ll_5 what leads to total costs of 17.

[2] https://sourceforge.net/projects/lpsolve/

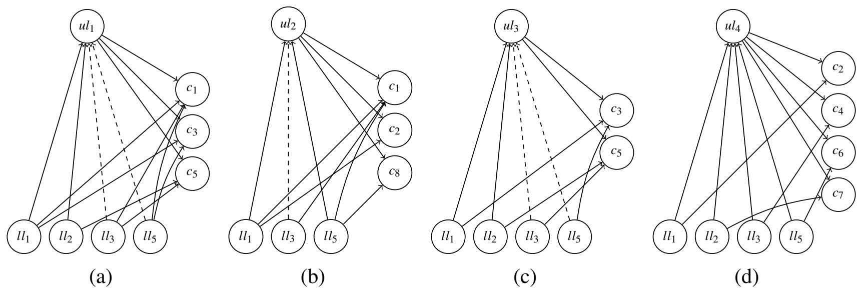


Fig. 5: Solution With Respect to Minimal Lower Layer Element Costs.

	ll_1	ll_2	ll_3	ll_4	ll_5
Assigned Cost	5	3	8	4	1

Tab. 2: To Lower Layer Elements Assigned Costs.

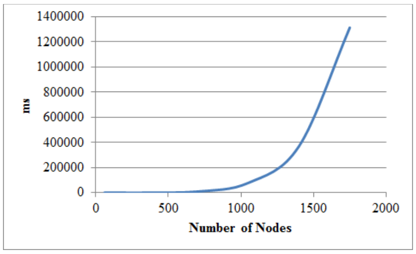
Does the Approach Scale?

We performed a series of experiments to evaluate if the proposed optimization approach is applicable for realistic industry-sized EAs. To this end, we randomly generated graphs consisting of 60 to 1750 nodes and measured the execution time needed to propose a solution for three optimization scenarios. Each generated graph is spit into two layers and the linking capabilities.

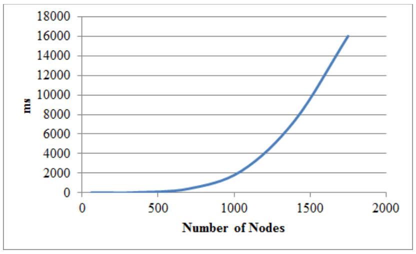
The obtained results are depicted in Fig. 6. The maximum execution time to compute a solution for the minimal coupling optimization scenario applied on a 1750 node graph was nearly 22 minutes (cf. Fig. 6 (a)). In contrast, applied on the same graph the optimizations regarding the minimal amount of lower layer elements and lower layer elements minimal costs took only 16 (cf. Fig. 6 (b)) respectively 5 seconds the longest (cf. Fig. 6 (c)). These remarkable differences can be explained by the fact that the constraints in the minimal coupling optimization scenario are strongly based on the relations between the layer elements and the capabilities. As each layer element is linked to several capabilities, the number of constraints grows faster compared to the other both optimizations scenarios.

The three scenarios have in common that the solution space is based on the cross product between the nodes of the two layers. Therefore, the execution time is growing exponentially in all three scenarios.

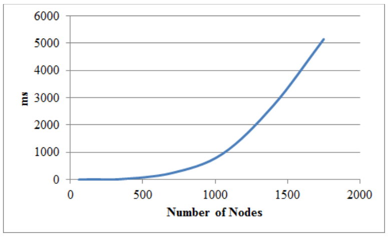
Unfortunately, only less data regarding the size of realistic EAs is available (Schoonjans reports on an EA consisting of 108 nodes [Sc16], Lagerstrom on one consisting of 407 nodes [La13]). Although the EA of one of our cooperation partners contains approximately 6000 nodes, the maximum number of elements which can be taken in account for a specified



(a) Minimal Coupling



(b) Minimal Lower Layer Element Amount



(c) Minimal Lower Layer Element Costs

Fig. 6: Execution Time Consumption

layer is below 500. Unfortunately, as no elements of this EA can be assigned to capabilities, we could not run our experiments on this EA as well.

Assuming that ordinary EAs are not significantly bigger, the execution time needed for all three optimization scenarios will be acceptable, especially because these optimizations need not to be applied frequently.

4 Conclusion

IT has become more important for business during the last decades and it changes commercial models dramatacly, which rises also the need for IT/business alignment. One way to ensure the IT/business alignment is EAM. To support EAM different techniques are feasible. In this paper we presented a technique to optimize the EA with focus on the relations between two adjacent layers.

Therefore, we introduced so called capabilities which are needed by the elements of the upper layer and are realized by the elements of the lower layer. We translated these construct into a LIP and searched for the optimal assignment between the elements of both layers. If we visualize these constraint, we see that the LIP has to ensure the existence of triangles.

Those triangles make it easier to give e.g. managers an insight into the LIP, because the constraints become more vivid and, therefore, easier to understand. This leads to a higher acceptance of the solution compared to existing solutions and, consequently, rises the chance to apply it.

Moreover, we presented a mapping between the elements of our technique and the widely accepted ArchiMate notation to enable organizations applying our approach. This mapping enables an optimization between the business and the application layer as well as between the application and the technology layer. Therefore, we identified the possible candidates for upper layer, lower layer, and capability elements. Additionally, we could show that our approach solves problems of a realistic size in appropriate time and, thus, is applicable to real world problems.

With respect to existing solutions, there are several possible extensions to our approach. First, we do not take the as-is state of the EA into account. But changes to the EA, as may suggested by our technique, cause transition costs, which can decrease the savings of the optimal solution. Second, we can assume different points in time with different costs and profit to predict the optimal time for each change. Third, we assume no dependencies between our elements within one layer. But there are such dependencies in reality. Therefore, our model should be extended in this direction. Last, the model need to be evaluated in existing organization. Even to test if the suggested optimizations really lead to savings.

References

[Ap11] Applegate, David L.; Bixby, Robert E.; Chvatal, Vasek; Cook, William J.: The traveling salesman problem: a computational study. Princeton university press, 2011.

[AW09] Aier, Stephan; Winter, Robert: Virtual Decoupling for IT/Business Alignment – Conceptual Foundations, Architecture Design and Implementation Example. Business & Information Systems Engineering, 1(2):150–163, 2009.

[AWW12] Aier, Stephan; Winter, Robert; Wortmann, Felix: Entwicklungsstufen des Unternehmensarchitekturmanagements. HMD - Praxis der Wirtschaftsinformatik, 284(49):15–23, 2012.

[BL08] Buhalis, Dimitrios; Law, Rob: Progress in information technology and tourism management: 20 years on and 10 years after the Internet: The state of eTourism research. Tourism Management, 29(4):609–623, 2008.

[Di70] Dinic, E. A.: Algorithm for solution of a problem of maximum flow in a network with power estimation. Soviet Math. Doll., 11(5):1277–1280, 1970.

[FF56] Ford, Lester R.; Fulkerson, Delbert R.: Maximal flow through a network. Canadian journal of Mathematics, 8(3):399–404, 1956.

[Fr10] Franke, Ulrik; Holschke, Oliver; Buschle, Markus; Rake-Revelant, Jannis; Närman, Per: IT Consolidation: An Optimization Approach. In: 14th IEEE International Enterprise Distributed Object Computing Conference Workshops. 2010.

[Gi12] Giakoumakis, Vassilis; Krob, Daniel; Liberti, Leo; Roda, Fabio: Technological architecture evolutions of information systems: Trade-off and optimization. Concurrent Engineering, 20(2):127–147, 2012.

[HR55] Harris, T. E.; Ross, F. S.: Fundamentals of a method for evaluating rail net capacities. DTIC Document, 1955.

[III11] ISO; IEC; IEEE: , Systems and software engineering – Architecture description, 01.12.2011.

[La04] Lankhorst, Marc M.: Enterprise Architecture Modelling - The Issue of Integration: Enterprise Modelling and System Support. Advanced Engineering Informatics, 18(4):205–216, 2004.

[La13] Lagerström, Robert; Baldwin, Carliss; MacCormack, Alan; Dreyfus, David: Visualizing and Measuring Enterprise Architecture: An Exploratory BioPharma Case. In (Grabis, Janis; Kirikova, Marite; Zdravkovic, Jelena; Stirna, Janis, eds): The Practice of Enterprise Modeling: 6th IFIP WG 8.1 Working Conference, PoEM 2013, Riga, Latvia, November 6-7, 2013, Proceedings, pp. 9–23. Springer, Berlin, Heidelberg, 2013.

[LW01] Landsbergen Jr., David; Wolken Jr., George: Realizing the Promise: Government Information Systems and the Fourth Generation of Information Technology. Public Administration Review, 61(2):206–220, 2001.

[OS00] Oliner, Stephen D.; Sichel, Daniel E.: The Resurgence of Growth in the Late 1990s: Is Information Technology the Story? FEDS Working Paper, (20), 2000.

[Pl07] Plazaola, Leonel; Flores, Johnny; Silva, Enrique; Vargas, Norman; Ekstedt, Mathias: An approach to associate strategic business-IT alignment assessment to enterprise architecture. In: Fifth Conference on Systems Engineering. 2007.

[PS04] Pereira, Carla Marques; Sousa, Pedro: A Method to Define an Enterprise Architecture Using the Zachman Framework. In: Proceedings of the 2004 ACM Symposium on Applied Computing. ACM, New York, NY, USA, pp. 1366–1371, 2004.

[Ro94] Rood, M. A.: Enterprise Architecture: Definition, Content, and Utility. In: Proceedings of the Third Workshop on Enabling Technologies. IEEE, New York, NY, USA, pp. 106–111, 1994.

[Sc16] Schoonjans, Anthony: Social Network Analysis techniques in Enterprise Architecture Management. PhD thesis, Ghent University, Ghent, 2016.

[SM96] Sanchez, Ron; Mahoney, Joseph T.: Modularity, flexibility, and knowledge management in product and organization design. Strategic Management Journal, 17(S2):63–76, 1996.

[ST03] Sundarraj, R.P; Talluri, Srinivas: A multi-period optimization model for the procurement of component-based enterprise information technologies. European Journal of Operational Research, 146(2):339–351, 2003.

[VSM10] Vodanovich, Shahper; Sundaram, David; Myers, Michael: Research Commentary—Digital Natives and Ubiquitous Information Systems. Information Systems Research, 21(4):711–723, 2010.

[WF06] Winter, Robert; Fischer, Ronny: Essential Layers, Artifacts, and Dependencies of Enterprise Architecture. In: 10th IEEE International Enterprise Distributed Object Computing Conference Workshops. IEEE, New York, NY, USA, pp. 30–38, 2006.

[Za87] Zachman, J. A.: A Framework for Information Systems Architecture. IBM Systems Journal, 26(3):276–292, 1987.