# INTEGRATING AN ENTERPRISE ARCHITECTURE USING DOMAIN CLUSTERING

Aier, Stephan, University of St.Gallen, Müller-Friedberg-Stasse 8, 9000 St.Gallen, Switzerland, stephan.aier@unisg.ch

Schönherr, Marten, Berlin University of Technology, Franklinstrasse 28/29, 10587 Berlin, Germany, mschoenherr@sysedv.tu-berlin.de

## Abstract

*Enterprise Architecture (EA) in the context of enterprise engineering addresses aspects of developing, improving and integrating organizations. The paper introduces an approach to EA proposing Integration Concepts (IC) to reconcile changing business process requirements and information systems. Being process-driven and supporting integration issues the chosen IC is a Service Oriented Architecture (SOA). Therefore the contribution aims at developing a methodology to support service engineering by defining architectural domains in an EA. The paper shows the need for methods in the field of domain engineering supporting the design of a SOA. The main contribution of the paper is an algorithm based modelling approach and a methodology to support service domain clustering. The clustering algorithms are using a model considering business processes, information systems and information system interfaces. The algorithm adopts network-centric approaches used in the field of social network analysis to define and/or identify service domain clusters in complex scenarios. The paper summarizes a case study in a globally operating company and closes with a conclusion. The paper is organized by chapters addressing context, objective, approach, case, results and lessons learned.*

*Keywords: enterprise architecture, integration concepts, process oriented integration, domain engineering, service domain clustering.*

## 1 CONTEXT: ENTERPRISE ARCHITECTURE AND ENTERPRISE INTEGRATION

Enterprise engineering methodologies describe the process of enterprise integration and enterprise modelling (ISO, 2000 p. 6). The following chapter gives a short introduction how we have developed an enterprise architecture approach covering integration of business processes, information systems and information system interfaces regarding to the mentioned understanding of Enterprise Engineering.

In a few words architecture can be defined as an abstract and holistic concept of structures and patterns considering planning aspects (Bass et al., 2003 p. 19; Winter & Fischer, 2006). Architectures are generally results of planning efforts and offer by definition a master plan supporting holistic implementation for future actions. These universal characteristics can be used for planning and designing of enterprise structures and strategies too. Furthermore enterprise architecture considers organizational, technical and psycho-social aspects for planning and building information systems (IS) in a socio-technical manner. This contribution particularly focuses on organizational and technical dimensions of EA. Therefore we use the terms *organizational architecture* and *IT architecture* (figure 1, see also Winter & Fischer, 2006; Williams et al. 2001, S. 27).

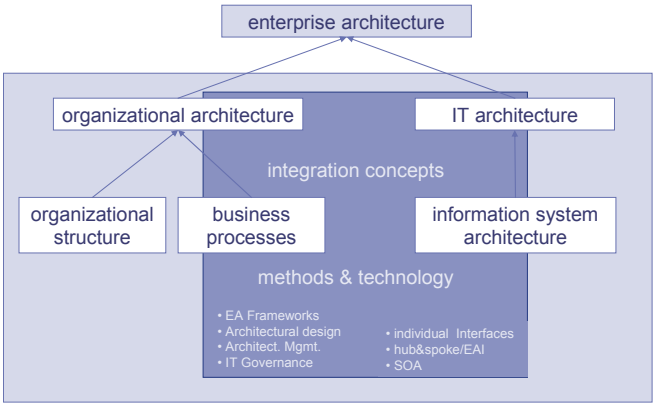


*Figure 1. Enterprise Architecture, see (Aier & Schönherr, 2006c)*

Organizational architecture contains all non-technical elements of the EA and is best compared with the so called instrumental understanding of organizations which covers all general explicit regulations to define the operational and organizational structure. Accordingly we differentiate the organizational architecture in organizational structure and business processes. On a par with the organizational architecture is the IT architecture which contains all technical elements of the EA. In particular IT architecture covers the IS which are described with their own architecture: the IS architecture. Both architectures organizational and IT architecture will be considered being equivalents but observed separately to accommodate the fact that both architectures are extremely relevant for the organization's efficiency and unfortunately do have complex interdependencies to each other.

Integration concept as middleware, enterprise application integration and service oriented architectures have been discussed, implemented and operated for many years. The following chapter will show the papers objective to cluster domains in an EA supporting service engineering aspects.

After all, designing and deploying integration concepts require both methodology and technology. This contribution will focus on aspects of methodology (for aspects of technology see Aier & Schönherr, 2006c). The recently discussed prominent approach to this issue is service oriented architecture (SOA). The paradigm of SOA is a distributed integration infrastructure (Aier & Schönherr, 2006c). One of the key benefits is a major challenge as well SOA's integration level. SOA not only aims at systems integration at a primarily technical level, but on business process integration driven by business requirements and implemented by utilizing information technology. To sum it up it can be said that SOA is major element and paradigm to design enterprise architecture. Therefore the following section will deal with the need for and the difficulties of managing services in a SOA. Thereafter we will show how clustering algorithms can be employed to derive services domain clusters from enterprise architectures. Eventually we will discuss methodologies for applying the algorithms proposed in practice.

# 2 OBJECTIVE: ARCHITECTURAL DOMAINS AND SERVICE ENGINEERING

By definition SOA delivers not just concepts for connecting elements of the IT architecture but reconcile IT and business processes. Both integration aspects are already considered in the technical definition of SOA which describe a business process driven IT integration. Therefore SOA could serve as a mediator between different elements of an EA.

Both scientists and practitioners emphasize the potential of SOA especially by reconciling business requirements and IT infrastructures as stated in the definition of enterprise architecture above by using integration concepts. Nevertheless there is the need for finding a stringent terminology hence common understanding used by the majority of the SOA community. Definitions range from a solely

technology driven approach to a new management school approach on how to run the whole enterprise.

"[A SOA is] a set of components which can be invoked, and whose interface descriptions can be published and discovered." (Gold et al., 2004 p. 72) Gold et al. mainly consider technological aspects focusing on standardized interface descriptions. Additionally McCoy and Natis taking into account aspects of stakeholder, granularity, reuse and agility (McCoy & Natis, 2003 p. 1). In the context of existing software systems and the introduction of SOA as a new overall enterprise architecture integration paradigm, issues as management and optimization need to be addressed too: "SOA is the concept of service-enabling new and existing software; linking internal and external service-enabled software systems; and implementing an enterprise wide infrastructure to enable, manage, and optimize services use and interaction" (New Rowley Group, 2003, see also Lubblinsky & Tyomkin, 2003; Roth, 2003)

Aside from primary SOA terminology many authors have a common understanding of secondary characteristics. Summed up these are the distributed manner of SOA, the aspect of combining (orchestration of) rich software components (services), loose coupling of applications using services and the standardization of interface descriptions (Sleeper & Robins, 2002; Lubblinsky & Tyomkin, 2003; Weinreich & Sametinger, 2001). To summarize the relevant issues Lubinsky and Tyomkin, 2003 highlight the business process driven integration and therefore derive the following three main aspects of a SOA:

- service descriptions
- business processes
- service (lifecycle) management

Because of complexity especially in large organizations the solely technical view on SOA is not sufficient to successfully implement SOA. Methodological aspects need to be considered too. The differentiation between technical and methodological issues is constantly discussed (Hagel & Brown, 2001; Gisolfi, 2001; Kirtland, 2001; Schelp & Winter, 2007).

Concerning service management as a term which summarizes methods to design and run SOA the following issues are relevant in the design time hence the phase when service characteristics as granularity and reuse are considered. Aside from basic service characteristics as mentioned, a service management defines a service lifecycle mainly to avoid service redundancy.

Due to reasons of complexity a methodology needs to be introduced to support early phases in the service lifecycle, the design time of a SOA. In the context of a SOA methodology terms as service management, domain engineering, governance, maturity and roadmaps can be found (IBM Corporation, 2005; Sonic, 2006). The most generic approaches can be found in the field of domain engineering (SEI, 2004; Open Group, 2006). The Domain Engineering introduced by the Carnegie Mellon SEI differentiates the following three activities (SEI, 2004):

- domain analysis
- domain design
- domain implementation

In the context of service definition the domain analysis needs to be considered. Output of a domain analysis is a domain model representing relevant features used for the specific context. To derive a domain model SEI proposes to use the context analysis defining the extent of a domain, the domain modelling providing a description of the relevant issues and the architecture modelling creating the architectural artefacts (SEI, 2004).

Adopting the general methodology of SEI to the requirements of a SOA especially in the context of an architectural migration from existing IT architectures to SOA a domain model needs to be created. A service domain can be described as a specific modelling view that consists of modularly defined functionality which is necessary to support business processes and the underlying basic data. Within dedicated IT projects, these requirements have to be implemented.

The following chapters describe methodologies, tools and algorithms to define service domains based on the generic SEI approach of a domain analysis, clustering existing enterprise architectures hence business processes, information systems and information system interfaces.

# 3 APPROACH: DOMAIN ENGINEERING BASED ON GRAPH CLUSTERING

As described above the approach of enterprise architecture is addressing integration concepts to reconcile volatile business process requirements and IT aspects. In this contribution we will focus on Service Oriented Architectures (SOA)

For clustering enterprise architectures first of all a model of the respective architecture is needed. Minimally the model should include the following elements:

- Business processes, that means the consecutive activities (tasks) and their relationships,
- IT systems,
- the usage of IT systems along a process,
- and the interrelationships and interfaces among IT systems along a process.

Such an enterprise model can be considered a graph often called network, too. In the following sections we will give a short introduction to graph theory and algorithms for graph partitioning. Thereafter we will introduce our implementation of a software system implementing those algorithms for enterprise architectures.

A graph consists of vertices $V$ and edges $E$. All elements of our model (activities and IT systems) can be considered vertices and their relations can be considered edges. If a connection between two IT systems is used several times along a process the model will have several edges between two vertices. These edges may also be combined into a single edge with the weight $w$. In this case the graph will be called weighted graph.

A network with $n$ vertices is represented by an $n \times n$ adjacency matrix $A$ with elements

$$A_{ij} = \begin{cases} 1 & \text{if the vertices } i \text{ and } j \text{ are connected} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

In a weighted graph $A_{ij}$ represents the weight of an edge between the vertices $i$ and $j$ (Newman, 2004).

The partitioning of such graphs into several modules is called clustering while a cluster consists of elements that are all similar to each other in some way (O'Madadhain et al., 2005). In our case of an enterprise architecture similarity means that vertices have a common subset of neighbours and are dependent from each other in some way. Typically this is a business activity which depends on the availability of an IT system.

Clustering approaches have a certain tradition in the analysis of social networks (Wasserman & Faust, 1999; Scott, 2005). Girvan/Newman proposed a clustering algorithm to identify communities in social networks (Girvan & Newman, 2002). Such a network consists of individuals (vertices) who know each other and thus have a relationship (edge). A community is defined by a number of people who know each other.

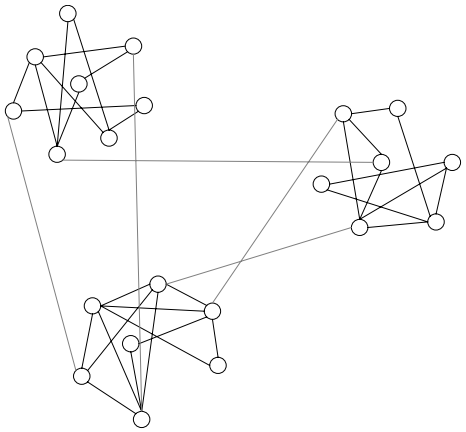

*Figure 2. A network with three communities, see (Girvan & Newman, 2002)*

Based on the analysis of shortcomings of existing clustering algorithms Girvan and Newman developed a "betweenness" algorithm. The basic idea of their algorithm is to remove the edges that are most in between in a network. The remaining vertices connected to each other form the communities. Therefore they generalized Freeman's betweenness (Freeman, 1977) to edges and defined the edge betweenness of an edge as the number of shortest paths between pairs of vertices that run along it. The shortest paths between communities run along only a few edges. That it is why these edges will have a rather high edge betweenness. By removing these edges one can separate the communities. The algorithm valid for a weighted network is the following (Newman, 2004):

1. Calculate the betweenness for all edges in the network.
2. Divide the betweenness by the weight of the respective edge.
3. Remove the edge with the highest resulting betweenness.
4. Recalculate betweennesses for all edges affected by the removal.
5. Repeat from step 3 until no edges remain.

In various papers they proved the performance of their algorithm in determining communities in social networks (Girvan & Newman, 2002; Newman, 2004; Newman & Girvan, 2004).

On an abstract level the problem of identifying modules in enterprise architectures is identical with the problem described for social networks. By applying the algorithm on the enterprise architecture model we identify appropriate modules as candidates for service domains in a SOA. Since one has to repeat the steps three to five for the weighted network several times it is also possible to identify a service domain hierarchy and eventually services. In every run through the algorithm, already identified modules will be further separated.

One of the important questions is when to stop the algorithm practically. Because it obviously does not make any sense to run through the algorithm as stated until no edges remain. This results in *n* modules which equals the number of vertices. So the question is when is a "good" modularity reached? Girvan and Newman propose the modularity *Q* as an indicator for the quality of clustering results (Newman & Girvan, 2004).

Therefore they calculate the fraction of edges that fall within a module.

$$\frac{\sum_{ij} A_{ij}\delta(c_i,c_j)}{\sum_{ij} A_{ij}} = \frac{1}{2m}\sum_{ij} A_{ij}\delta(c_i,c_j) \tag{2}$$

Where $c_i$ is the module the vertex *i* belongs to. The function $\delta(u,v)$ is 1 if u = v and 0 otherwise, and $m = \frac{1}{2}\sum_{ij} A_{ij}$ is the number of edges in the graph. If the degrees *k* of vertices (the degree $k_i$ of a vertex *i* is defined by the number of vertices connected with *i*) in a network are preserved but otherwise

connect vertices together at random, then the probability of an edge existing between vertices $i$ and $j$ is $k_i k_j / 2m$, where $k_i$ is the degree of vertex $i$. Thus the modularity $Q$, is given by

$$Q = \frac{1}{2m} \sum_{ij} \left[ A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j) \tag{3}$$

Values for Q range between 0 and 1. A value of 0 indicates a poor clustering result while 1 is a perfect cluster but a rather theoretical value. Usually values between 0.3 and 0.7 indicate good values for realistic examples (Newman & Girvan, 2004).

# 4 CASE: MODELLING ENTERPRISE ARCHITECTURE

After analyzing EA modelling tools we decided to implement a prototype applying the described algorithm on EA models based on a meta-model considering existing business processes, IT systems, interfaces, and systems integration on a business process level.

The performance of the clustering algorithm for enterprise architecture models has been verified on a number of specially prepared test cases discussed in (Aier, 2006a).

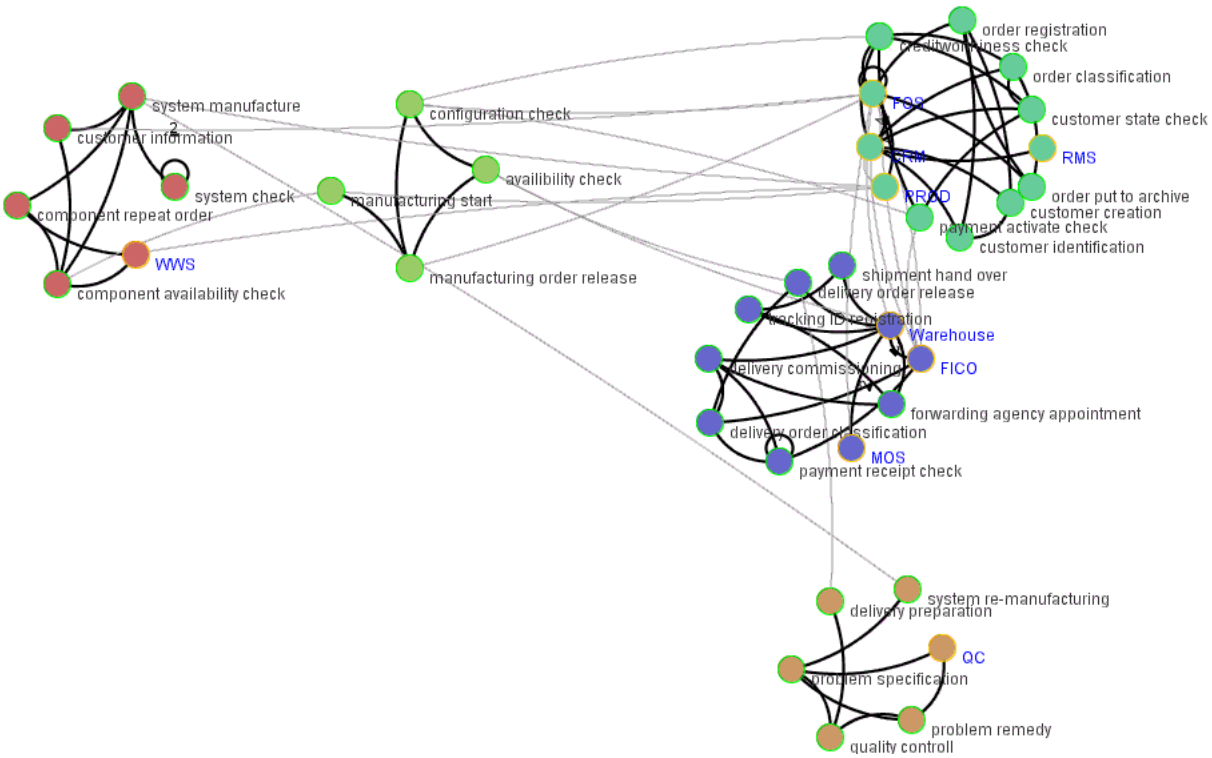


*Figure 3. Screenshot of EA Builder model transformed in a graph, 20 edges removed resulting in 5 clusters, grouped view*

The first step is to model relevant processes, IT systems, and interfaces in the as-is state. The approach extends the event driven process chain especially incorporating business process oriented systems integration. After modelling the as-is state the clustering algorithm supports an architect in order to design appropriate clusters in an EA. The tool and the methodology hence simple have been used in an extensive case in a globally operating telecommunications supplier. In the case study we modelled about 100 IT systems, 400 activities in business processes as well as the events, operators, and various relations on different levels of abstraction. Additionally we applied the clustering algorithm to identify domain clusters. Figure 3 shows an example made anonymous from the case where five clusters were derived removing 20 edges by the algorithm driven tool in the as-is architecture.

# 5 RESULTS AND LESSONS LEARNED

The application of the clustering algorithm on the case study resulted in 44 clusters identified. Afterwards the clusters were discussed with the process and IT owners concerned. They described the clusters as meaningful from a business perspective. The process of discussing the resulting clusters has been a very helpful exercise in order to understand the interplay of business processes and IT systems. However, in some cases the process and IT owners were surprised that a certain element was placed in a cluster they would normally not have assigned it to. This may be interpreted as a hint, that the clustering approach reveals the actual structure of an EA and not the structure documented in organizational charts etc. Moreover the discussion of clusters also revealed processes which were not IT supported or processes which were redundantly supported by more than one IT system.

However, the work presented here is work in progress obviously. For instance the approach covers the analysis of an as-is architecture only and our method is very straight forward so far. It is necessary to extend the method in order to construct a to-be architecture from the clustering results. Therefore it is necessary to include further EA artefacts as for example information objects, products etc. (Winter & Fischer, 2006). One of the advantages as well as a disadvantage of our approach is its simplicity. Removing all the semantics from an EA model in order to cluster the resulting network may be problematic for answering questions in the area of EA design, service design etc. Going one step further we would have to identify relevant attributes of the EA elements and include them in the clustering itself. Finally it would be interesting to compare the performance of other clustering algorithms, approaches from the database community like (Heiland & Kruck, 1993) or approaches from the software engineering community like (Girard & Koschke, 1997). The evaluation of the proposed improvements should be done in a design science tradition (Hevner et al., 2004) by employing the improved algorithms and methods in case studies.

## References


Aier, S. (2006a) How clustering enterprise architectures helps to design service oriented architectures. In Proceedings of the IEEE International Conference on Services Computing (SCC'06), pp. 269–272, IEEE Computer Society, Los Alamitos, CA, USA, Chicago, USA.

Aier, S. (2006b) Public information on ea builder on the internet. http://www.ea-builder.com.

Aier, S. and Schönherr, M. (2006c) Evaluating integration architectures – a scenariao-based evaluation of integration technologies. In Trends in Enterprise Application Architecture, Revised Selected Papers (Draheim, D. and Weber, G., Eds), Lecture Notes in Computer Science, Vol. 3888/2006, Springer, Berlin, Heidelberg, pp. 2–14.

Bass, L., Clements, P. and Kazman, R. (2003) Software architecture in practice. Pearson Education Inc., Boston.

Duffy, J. (2001) It/business alignment: Delivering results. CIO magazine, http://www.cio.com/analyst/123101_idc.html.

Freeman, L. C. (1977) A set of measures of centrality based upon betweenness. Sociometry 40, pp. 35–41.

Girard, J.-F. and Koschke, R. (1997) Finding Components in a Hierarchy of Modules: a Step towards Architectural Understanding. icsm, 13th International Conference on Software Maintenance (ICSM'97), p. 58.

Girvan, M. and Newman, M. E. J. (2002) Community structure in social and biological networks. Proceedings of the National Academy of Science 99 (12), pp. 7821–7826.

Gisolfi, D. (2001) Web services architecture: Part 1- an introduction to dynamic e-business. http://www-106.ibm.com/developerworks/webservices/library/ws-arcl/.

Gold, N., Knight, C., Mohan, A. and Munro, M. (2004) Understanding service-oriented software. IEEE Software (2), pp. 71–77.

Hagel, J. and Brown, J. S. (2001) Your next it strategy. Harvard Business Review 79 (9), pp. 105–113.

Heilandt, T. and Kruck, P. (1993) Ein algorithmisches Verfahren zur Bewertung und Verdichtung von Entity-Relationship-Modellen. Informatik, Forschung und Entwicklung, 8, pp. 197-206.
Hevner, A.; March, S.; Park, J. and Ram, S. (2004): Design Science in Information Systems Research. MIS Quarterly, 28 (1), pp. 75–105.
Ibm Corporation (2005), http://www-128.ibm.com/developerworks/webservices/library/ws-soa-simm/.
Iso (2000) Industrial automation systems – requirements for enterprise reference architectures and methodologies. 15704, ISO, p. 43.
Kirtland, M. (2001) A platform for web services. Microsoft developer network. http://msdn.microsoft.com/library/default.asp?/en-us/dnwebsrv/html/websvcs_platform.asp.
Lubblinsky, B. and Tyomkin, D. (2003) Dissecting service-oriented architectures. Business Integration Journal, pp. 52–58.
Luftman, J. (2003) Measure your business-it alignment. Optimize Magazin, http://www.optimizemag.com/article/showArticle.jhtml?articleId=17701026.
Mccoy, D. and Natis, Y. (2003) Service-oriented architecture: Mainstream straight ahead. Gartner Research.
New Rowley Group (2003) Building a more flexible and efficient it infrastructure – moving from a conceptual soa to a service-based infrastructure. http://www.newrowley.com/reseach.html.
Newman, M. E. J. (2004) Analysis of weighted networks. Phys. Rev. E 70 (056131),
Newman, M. E. J. and Girvan, M. (2004) Finding and evaluating community structure in networks. Phys. Rev. E 69 (026113),
O'madadhain, J., Fisher, D., Smyth, P., White, S. and Boey, Y.-B. (2005) Analysis and visualization of network data using jung. Journal of Statistical Software, noch nicht erschienen.
Open Group (2006), http://www.opengroup.org/architecture/togaf8-doc/arch/p4/maturity/mat.htm.
Roth, P. (2003) Moving to a service based architecture. Business Integration Journal, pp. 48–50.
Schelp, J. and Winter, R. (2007) Towards a methodology for service construction. In Proceedings of the 40th annual hawaii international conference on systems sciences (hicss'07), jan 3-6 2006, hawaii, USA, IEEE Computer Society Press, Los Alamitos, CA, USA.
Scott, J. (2005) Social network analysis: A handbook. SAGE, London u.a.
Sei (2004), http://www.sei.cmu.edu/domain-engineering/.
Sleeper, B. and Robins, B. (2002) The laws of evolution: A pragmatic analysis of the emerging web services market. The Stencil Group, San Francisco.
Sonic (2006), www.sonicsoftware.com/soamm.
Wasserman, S. and Faust, K. (1999) Social network analysis: Methods and applications. Cambridge Univ. Press, Cambridge u.a.
Weinreich, R. and Sametinger, J. (2001) Component models and component services: Concepts and principles. In Component-based software engineering: Putting pieces together (Council, W. T. and Heinemann, G. T., Eds), pp. 22–64, Addison Wesley, Boston.
Williams, T. J., Li, H. and Rathwell, G. A. (2001) A Handbook on Masterplanning and Implementation for Enterprise Integration Programs. West Lafayette, Indiana: Purdue Laboratory for Applied Industrial Control. 2004, pera.net, Zugriff am 10.6.2004.
Winter, R. and Fischer, R. (2006) Essential layers, artifacts, and dependencies of enterprise architecture. In Edoc workshop on trends in enterprise architecture research (tear 2006), hong kong, 16.10.2006 (Ieee Computer Society, Ed), IEEE Computer Society, Los Alamitos, CA, USA.