Automating Enterprise Architecture Documentation using an Enterprise Service Bus

Journal:	*18th Americas Conference on Information Systems*
Manuscript ID:	AMCIS-1631-2012.R1
Submission Type:	Paper
Track:	Enterprise Architecture and Organizational Success < Enterprise Systems (SIGEntSys)

SCHOLARONE™
Manuscripts

Automating Enterprise Architecture Documentation using an Enterprise Service Bus

Markus Buschle
Industrial Information and Control Systems
KTH Royal Institute of Technology
Osquldas v. 12, SE-10044 Stockholm, Sweden
markusb@ics.kth.se

Mathias Ekstedt
Industrial Information and Control Systems
KTH Royal Institute of Technology
Osquldas v. 12, SE-10044 Stockholm, Sweden
mathias.ekstedt@ics.kth.se

Sebastian Grunow
Chair for Informatics 19 (sebis)
Technische Universität München
sebastian.grunow@tum.de

Matheus Hauder
Chair for Informatics 19 (sebis)
Technische Universität München
matheus.hauder@tum.de

Florian Matthes
Chair for Informatics 19 (sebis)
Technische Universität München
matthes@tum.de

Sascha Roth
Chair for Informatics 19 (sebis)
Technische Universität München
roth@tum.de

ABSTRACT

Currently the documentation of Enterprise Architectures (EA) requires manual collection of data resulting in an error prone, expensive, and time consuming process. Recent approaches seek to automate and improve EA documentation by employing the productive system environment of organizations. In this paper, we investigate a specific Enterprise Service Bus (ESB) considered as the nervous system of an enterprise interconnecting business applications and processes as an information source. We evaluate the degree of coverage to which data of a productive system can be used for EA documentation. A vendor-specific ESB data model is reverse-engineered and transformation rules for three representative EA information models are derived. These transformation rules are employed to perform automated model transformations making the first step towards an automated EA documentation. We evaluate our approach using a productive ESB system from a leading enterprise of the fashion industry.

Keywords

Enterprise Architecture (EA) management, model transformation, automated EA documentation, Enterprise Service Bus, SAP Process Integration.

INTRODUCTION

Trends such as globalization and frequent changing market requirements challenge an enterprise to reduce costs in order to gain profit. The management discipline of Enterprise Architecture (EA) is finding acceptance as an approach to align business and IT to gain strategic advantages over competitors by increasing flexibility of IT, identifying and realizing cost-saving potentials, increasing availability and failure tolerance, etc. (Weill and Ross 2009; Ross, Weill, and Robertson 2006; Zachman 1987). While classical software engineering approaches focus on details, EA management tries to convey a holistic view of the entire enterprise (Buckl, Matthes, Roth, Schulz, and Schweda 2010). Starting point of an EA endeavor commonly embraces the documentation of the current EA (Kurpjuweit and Winter 2007). As Enterprise Architecture commonly includes a large variety of business and IT artifacts as well as artifacts about their alignment, such documentation processes typically are regarded as time consuming and cost intensive (Farwick, Agreiter, Breu, Ryll, Voges, and Hanschke 2011; Farwick, Agreiter, Ryll, Voges, Hanschke, and Breu 2011; Department of Defense Architecture Framework Working Group & others 2007; Kaisler, Armour, and Valivullah 2005).

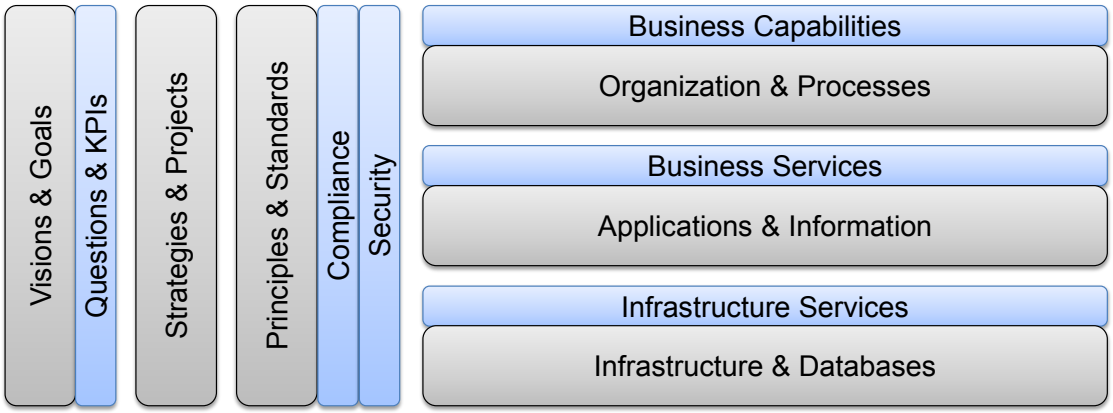

Figure 1. Holistic view on an Enterprise Architecture

Consequently, to document the EA, most enterprises are first concerned with the creation of a (formal) model specifying the information about the EA to be documented. These models also referred to as information models (Lee 1999) embrace concepts, relationships, constraints, rules, and operations to describe the EA formally. An analysis of literature reveals a considerable number of information models aimed at describing the EA information demand, e.g., ArchiMate (The Open Group 2009a, 2012), CySeMoL (Sommestad, Ekstedt, and Johnson. 2010), and planningIT (Alfabet AG 2012). Attempting to consolidate the different views (Buckl et al. 2010) on the EA information demands Figure 1 illustrates a compendious view on an EA. It starts from bottom-up with *infrastructure & databases*, e.g. networks, routers, server farms, etc. Those infrastructure elements are provided as infrastructure services to an upper layer, where *applications & information* are employed to realize *business services*. Business services are cross-grained services that build *business processes* executed by the *organization*. Services can be rearranged in order to create new business processes. *Business capabilities* describe core competencies enterprises offer, whereas the organizational structure is organized to efficiently execute business processes and function most effectively.

So-called cross-cutting aspects influence afore introduced layers. A common *vision* is used to derive *goals* that are measured in *KPIs* answering *questions*. On all introduced levels, *strategies* and *projects* drive change that is guided by corporate *principles* and *standards* with respect to external factors like *compliance* and *security*. In EA management, a primary goal is to meet 'the right' information demands of involved stakeholders. Those are manifold and could embrace the entire EA or parts thereof (cf. Figure 1). An analysis of approaches for gathering the EA information, e.g. the current state of the application landscape, reveals a high degree of manual effort, e.g. interviews with information stewards, resulting in an error-prone and time consuming task. With increasing requirements on flexibility, agility, etc. recent approaches are not able to meet current challenges, in particular since there is a constantly growing information volume and no chance to determine the right information at the right quality.

Recently, approaches for automating information gathering processes are proposed. Allowing for the existence of a lot necessary information in the operative IT, (Buschle, Holm, Sommestad, Ekstedt, and Shahzad 2011) and (Farwick, Agreiter, Ryll et al. 2011) propose to employ the IT of the infrastructure and database layer as an information source in order to avoid the expensive task of manual information collection. Nevertheless, even though these authors address this problem, only little research is done on the analysis of a potential information source.

An Enterprise Service Bus (ESB), as the nervous system of an enterprise, coordinating interactions between business applications and processes could contain suitable information. Our research approach shown in Figure 2 envisions a productive ESB for the automated EA documentation. The approach is evaluated at an enterprise in the fashion industry using SAP PI as ESB.

After revisiting related work, we follow the previously outlined approach starting with reverse-engineering the data model of SAP PI. We then compare this model with existing EA information models namely ArchiMate as a general approach, CySeMoL as an approach focusing on the *infrastructure & database* layer as well as planningIT, a widespread EA tool developed by alphabet. The model coverage for ArchiMate that can be automatically extracted from an existing SAP PI instance is highlighted next. The automatic EA documentation of information is realized with transformation rules using the Atlas Transformation Language (ATL). Finally, we briefly compare the ArchiMate model coverage with the achieved coverage of CySeMoL and planningIT.

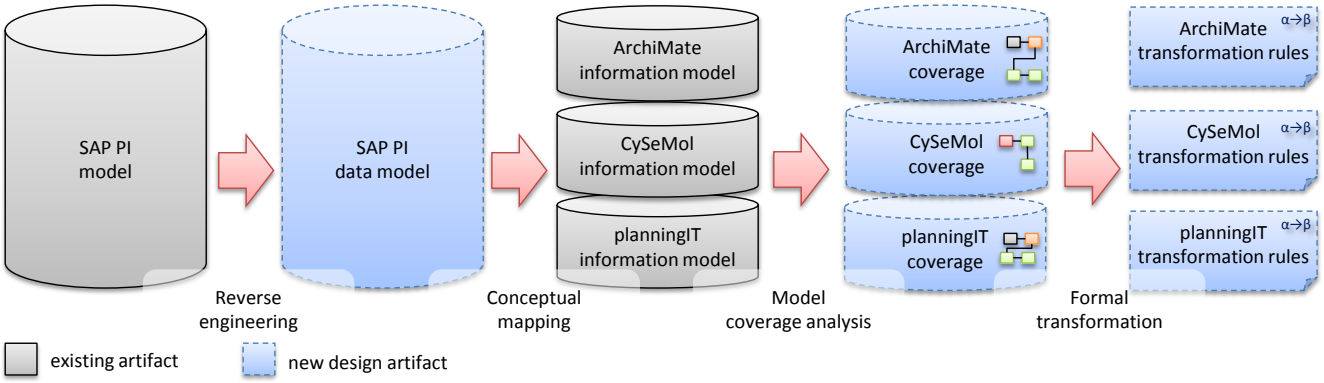

Figure 2. Approach for model transformations towards automated EA documentation

RELATED WORK

Despite the existence of several EA frameworks including inter alia *The Open Group Architecture Framework* (TOGAF) (The Open Group 2009b), *The Integrated Architecture Framework* (Wout, Waage, Hartman, Stahlecker, and Hofman 2010), and *Enterprise Architecture Planning (EAP)* (Spewak and Hill 1993) commonly the frameworks abstract from details on how to acquire and incorporate Enterprise Architecture knowledge from other sources (Buckl and Schweda 2009). Only few approaches addressing the documentation of the status quo can be found. However, the suggestions made are usually at a high abstraction level and limited to giving a few advices rather than concrete task descriptions. For instance, TOGAF suggests the usage of existing architecture definitions as a starting point, which, if necessary, can be updated and verified. In case such descriptions do not exist TOGAF advices to gather data in "*whatever format comes to hand*" (The Open Group 2009b) not providing guidelines for dealing adequately with information on Enterprise Architecture level.

A first idea for an automated tool-aided processes can be found in (Moser, Junginger, Brückmann, and Schöne 2009) proposing a set of EA process patterns, one of which is called *Automatic Data Acquisitions/Maintenance*. The authors present a process to automatically collect data from various sources subsequently converted into an EA information model instance. However, the considerations remain at a high abstraction level not detailing the mapping of collected data onto EA information. This also applies to (Farwick, Agreiter, Ryll et al. 2011). Apart from identifying requirements on a potential automated process, they develop a basic structure of an automated maintenance process comprising the collection of data as well as the propagation of changes. The introduced process is composed out of four different sub-processes including the addition of new information sources, the propagation of changes, the integration and consolidation of different data flows and finally the evaluation of the data by stakeholders. When it comes to the conversion of the extracted data to the internal EA information model (Farwick, Agreiter, Ryll et al. 2011) refer to "*a semantic mapping between the provided data, and the internal data structure*" without going further in-depth.

Up till now one publication could only be identified dealing with the transformation of extracted data to EA information. Buschle et al. (Buschle et al. 2011) use NeXpose (Rapid7 2012), a vulnerability scanner, aimed at determining weaknesses within a network automatically. Apart from security risks, the scanner collects information about the system landscape with focus on technology and application aspects subsequently converted into EA information. The transformation is realized by extracting elements from the NeXpose reports stored in XML files and mapping them to the corresponding CySeMoL elements in the second step. For instance, the data provided by NeXpose allow the authors to gather a great deal of information comprising inter alia the determination of services, installed software, and used operating systems. However, gaps exist above all in business fields, e.g., the determination of business objects and business processes as well as in application aspects, e.g., the collection of provided services and exchanged information. With regard to an application of ESB as an information source for EA documentation no publication, both in practice as well as in research, could be found tackling this subject. Accordingly, this is the first contribution in this area.

ANALYSIS AND FORMAL DESCRIPTION OF SAP PI DATA

Main purpose of an Enterprise Service Bus (ESB) is to manage the interaction between different software applications within an enterprise. Since we use SAP PI from our industry partner as a concrete implementation of an ESB, its main software components are shown in Figure 3. Based on SAP PI's database schema a data model is reverse-engineered in the first step.

In a second step, a study of the SAP documentation revealed the main elements of the system. Both steps are necessary to obtain the model coverage of the EA information models and derive a transformation method in order to automatically document the EA.

Subsequently these components are investigated in detail along with the data model shown in Figure 4. The data model only shows the most relevant entities while less important details are omitted from the figure.

System Landscape Directory

This component is the central source of information for the entire system landscape. Within the *system landscape directory* there is a distinction between *software products* and *software components*. The *software product* is, according to the official definition, a unit that can be delivered, is visible to the customer, is installable and can be renewed. A *software component* in turn represents the reusable and not directly installable components of a *software product*. While *software component* and *software product* represent installable software in its original state, *installed software product* and *installed software component version* model concrete installations which can be updated and altered with patches. A separation is made between *technical systems* representing *application systems* and *business systems* which are logical elements functioning as senders, receivers or both. In this context a *business system* is associated with exactly one *technical system*, so that changes to the technical infrastructure has no effect on the design elements but only the relationship between both types of systems (Nicolescu 2009).

Enterprise Service Builder

The *enterprise service builder* serves as central environment to design, create and maintain the interactions between applications (Nicolescu 2009). This includes the description of service interfaces in an independent design time representation of a service. In case the service is provided or expected from the environment a distinction is made between *inbound* (receiving messages) and *outbound interfaces* (sending messages). These services are composed of one or more operations. The structure of an operation is described, inter alia, by data types allowing the description of input and output messages. While the simple message processing is stateless, the *enterprise service builder* allows the definition of integration processes that are able to use the knowledge of messages that already have been preceded.

Integration Builder

The *integration builder* specifies the configuration of communication relationships at runtime by mapping the design information stored in the *enterprise service builder* to the actual execution environment (Nicolescu 2009). Messages are processed by *communication components* that are specialized to *integration processes*, *business systems* and *business components*. A *business component* describes an abstract unit which is usually used in business to business communication relationships in order to hide details of the s*ystem landscape*. In addition, sender as well as receiver agreements contain a reference to *communication channels* specifying the configuration of an adapter for a specific communication relation. Routing rules are determined by *receiver determinations* defining which receivers a message is sent to as well as i*nterface determinations* providing information about which interfaces a message is sent to. In both cases the source interface and source communication component are taken into account for the identification of the target elements. *Party* is a larger unit, especially involved in cross company processes (SAP 2009).

RELATING SAP PI DATA WITH EA INFORMATION DEMAND

The challenge of using existing IT runtime systems for EA documentation is to be able to extract useful Enterprise Architecture information. For that, the extent to which the EA information demand set up by selected information models is met by SAP PI is analyzed below. The aim is not only to compare SAP PI with a set of information models, but also to draw representative conclusions about the application potential of SAP PI in general. Consequently, the selection attempts to achieve a representative overview, including ArchiMate, CySeMoL, and planningIT. Despite differences with regard to ESB architectures, they offer similar functionality leading to conformity of the data content. Accordingly, the following considerations are supposed to provide a starting point for other ESB systems.

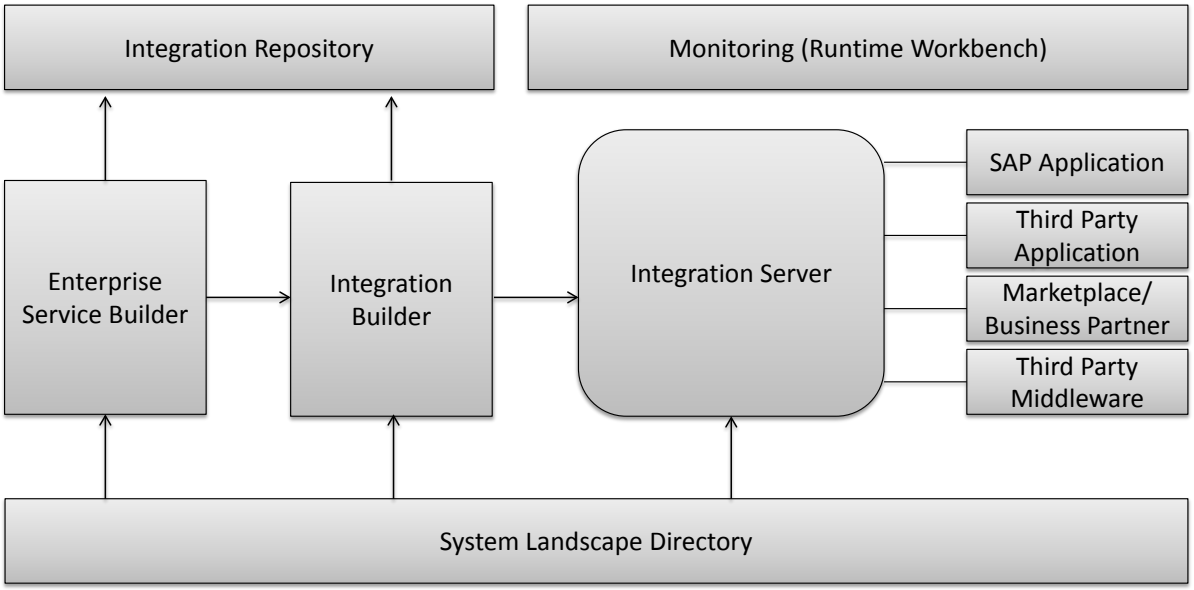

Figure 3. Overview of the different software components in the Enterprise Service Bus system SAP PI

ArchiMate is an independent modeling language maintained by The Open Group. Conforming with the general view of EA information content (Fischer, Aier, and Winter 2007), ArchiMate focuses on conveying a holistic view of the enterprise while intentionally abstracting from details. This global, comprehensive way of viewing demonstrates yet a first difference compared to SAP PI whose data are intended to detail the interactions and communications to allow automated processing. The mapping of SAP PI data content onto EA information models must therefore follow two principles, abstraction from details and combination of data leading to data on higher granularity levels.

Within ArchiMate the architecture of an organization is divided into three main layers (business, application, and technology) and categorized into three concepts (structural, behavioral, informational). The ArchiMate information model is shown in Figure 5 and is compared with the previously outlined SAP PI data model in Figure 4.

Business Layer

The previous, detailed analysis of SAP PI's data content already leads to the presumption that SAP PI as a productive system is only of limited value to determine business information. However, when considering business aspects of SAP PI, it emphasizes the idea of business information that is implicitly contained in its data. Reconstruction thereof is answered below.

The informational concepts, intended to link the operational side with business goals, are made up of information carriers (*business objects*, *products,* and *contracts*), information representation (*representation*), and elements describing their commutative goals (*meaning* and *value*). Considering the formal SAP PI data model, information about underlying, pursued goals is completely absent. In contrast, even though *business objects*, understood as "*a unit of information relevant from a business perspective*" (The Open Group 2012), are not directly included in SAP PI, *data types* can be seen as first indications on their existence. Following the SAP PI best practice data naming conventions *data types* are named after the corresponding *business object* they are supposed to implement (SAP 2009). In this connection, SAP PI *adapters* provide first glance about their *representation*, including technologies such as e-mail. Finally, *products*, a collection of *application* and *business services*, are one of ArchiMate's elements about which clues are included in SAP PI but the strong technical focus does not allow to reconstruct them unequivocally. On the one hand, SAP PI does not directly describe the provided services but only their way of access. On the other hand, an automated aggregation of services going together into products remains questionable.

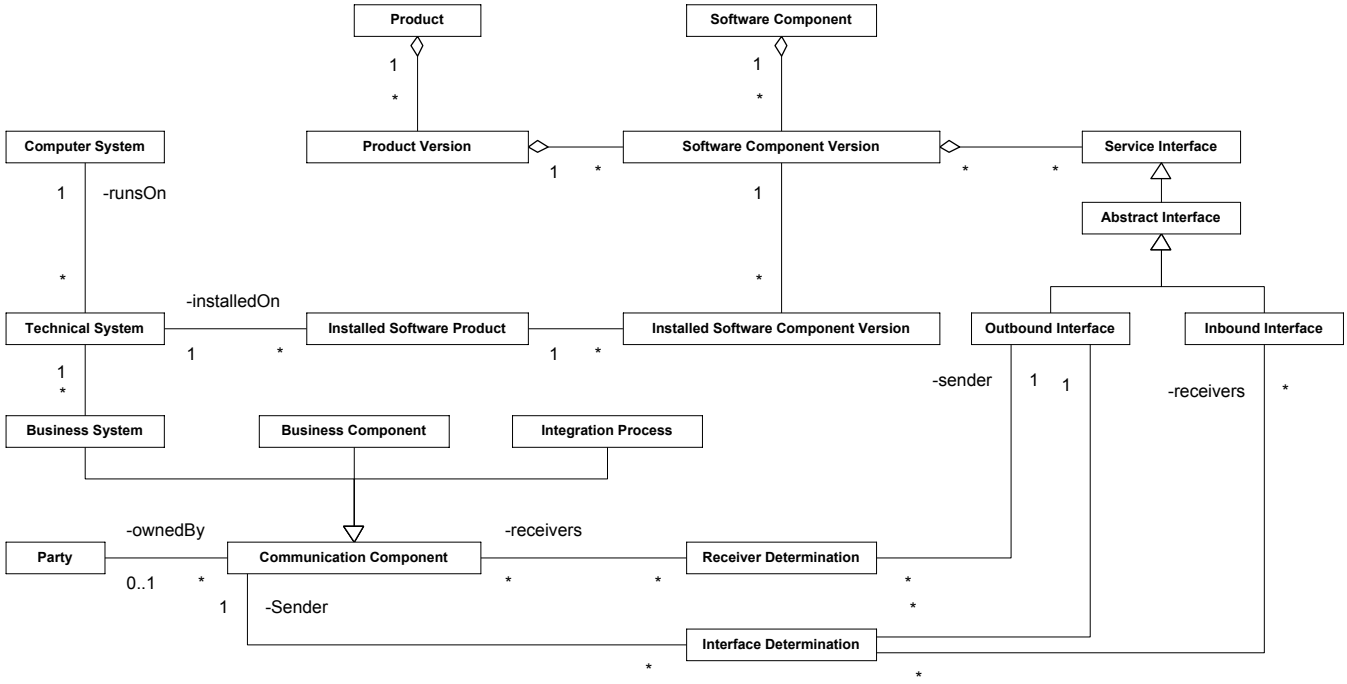

Figure 4. Reverse-engineered data model of the SAP PI Enterprise Service Bus

The point of access of provided functionality is modeled as *business interfaces* ranging from technologies such as mail to complex structure such as departments receiving and forwarding requests. While SAP PI is entirely suited regarding the former, information about complex business connections is obsolete. In contrast, *business actors* represent organizational units with the capability of performing behavior. Even though the corresponding SAP PI element, *party*, commonly refers to an organization and thus appears to be too coarse grained, it can be advantageous to interpret the definition of *business actor* more broadly, especially, when analyzing cross-company relationships.

With regard to behavioral elements a distinction is drawn between external (*business services*) and internal visible functionality (*business processes* and *business functions*). Defined as a piece of functionality offering added values to the environment, business services are not considered in SAP PI only describing the way of access in the form of technologies (see *business interfaces*) abstracting from functionality information. The description of internal behavior in ArchiMate ranges from a process view (*business process*) to a functional view (*business function*). A *business process* is understood as "*a unit of internal behavior or collection of causally-related units of internal behavior intended to produce a defined set of products and services*" (The Open Group 2012). The definition of *integration processes* comes closest differing, however, in the technical focus neglecting the aspect to follow an overall goal. Similar conclusions can be drawn for the other behavior elements.

To sum up, although SAP PIs elements are meant to implement business functionality a reconstruction of business information is commonly hindered by the strong technology focus. Even at this early stage the hypothesis can be proposed that this also applies to other productive systems to be used for an automated process. For instance, (Buschle et al. 2011) came to the same conclusion regarding NeXpose, primary providing technology information.

Application Layer

Central to the application layer is the *application component* specified as a "*modular, deployable, and replaceable part of a system*" (The Open Group 2012). While SAP PI introduces two similar concepts, *software components* and *software products*, *software components* are not meant to be deployable in contrast to *software products*. A temporary configuration of two or more *application components* aimed at performing a higher functionality is modeled as an *application collaboration*. A similar objective occurs at *integration processes* describing the coordinated message exchange between several *software products*. Accordingly, the *products* involved in the message exchange processes form an *application collaboration*. Access to the underlying services provided by *application components* as well as their groupings is modeled by *application interfaces*, broadly equivalent to SAP PI's *enterprise service interface*. The determination of which *application component*

invokes which *interface* is implicitly included in SAP PI's routing information (*receiver determination* and *interface determination*) defining the message exchange between *enterprise service interfaces* and *software products*.

Similar to the business layer, the behavioral concept makes a distinction between internal and external visible functionality. While internal functionality of *application components* remains invisible to an ESB, first indications on external visible functionality (*application services*) exist. In SAP PI no specific elements are intended to be used for describing behavioral information. Instead, such information is available rather indirectly in *interfaces* and the included descriptions of *operations*. However, even though the *operations* contain all service information, it is questionable whether the *operations* can be automatically aggregated to specify the *service* forming the combined functionality.

The informational concepts are made up by only one element, the *data object*, "*a coherent, self-contained piece of information suitable for automated processing*". All of these requirements are met by SAP PIs *data types*.

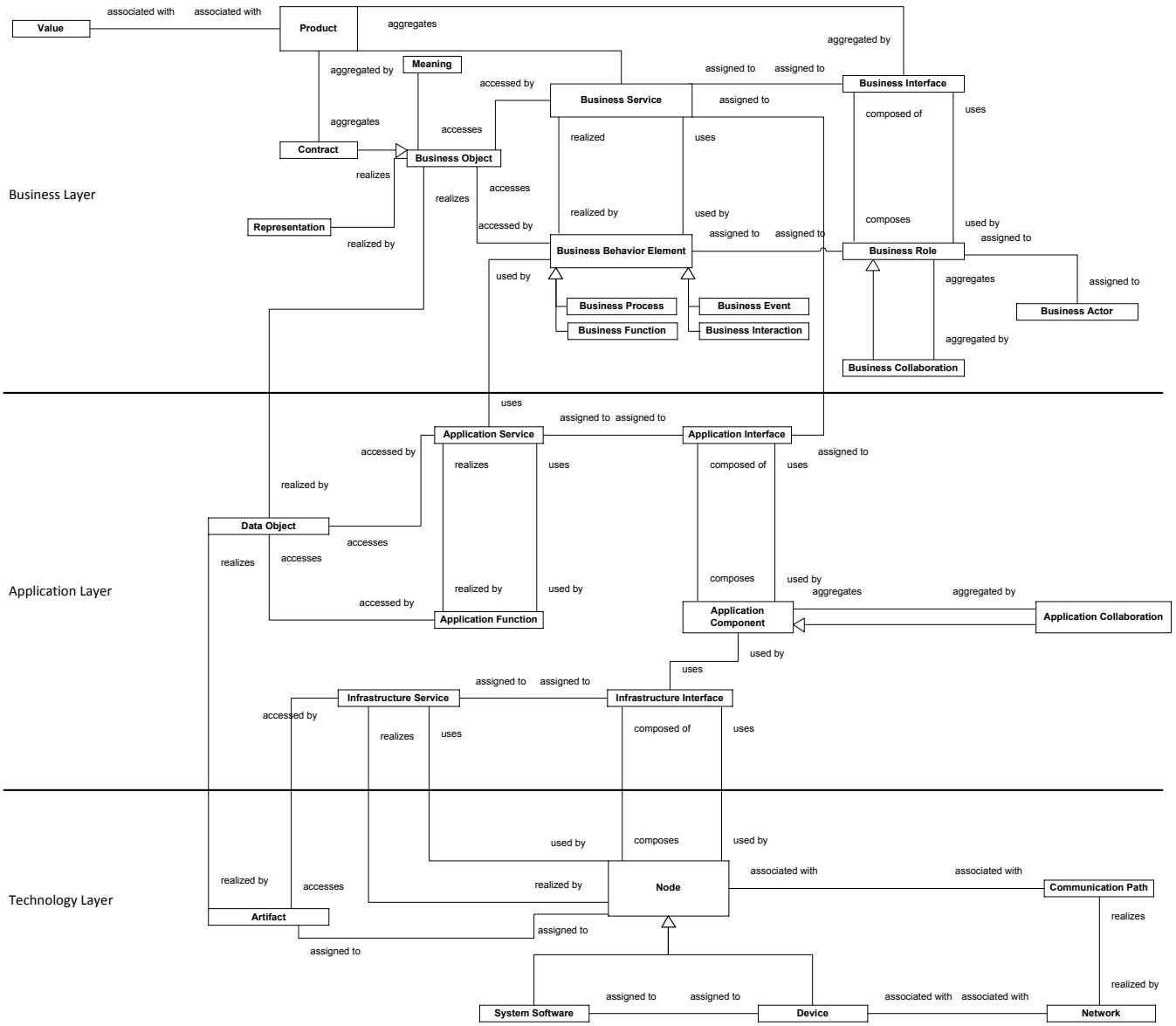

Figure 5. ArchiMate information model according to (The Open Group 2012)

Technology Layer

The technology layer comprises information about the underlying infrastructure. This begins with *node,* modeling a computational resource and thus corresponds to SAP PI's *computer system*. In contrast to the application layer, as the provided and needed *interfaces* of infrastructure components are not of importance for the coordination of applications, no information appears from the available data. Besides components, two types of relationships on different level of abstraction exist, *communication paths,* modeling a logical exchange relationship and *networks* referring to physical communicational medium. While the underlying physical mediums are abstracted in SAP PI each invocation of a service comprises two *communication paths*, one between the service client and SAP PI and the other one between SAP PI and the service provider.

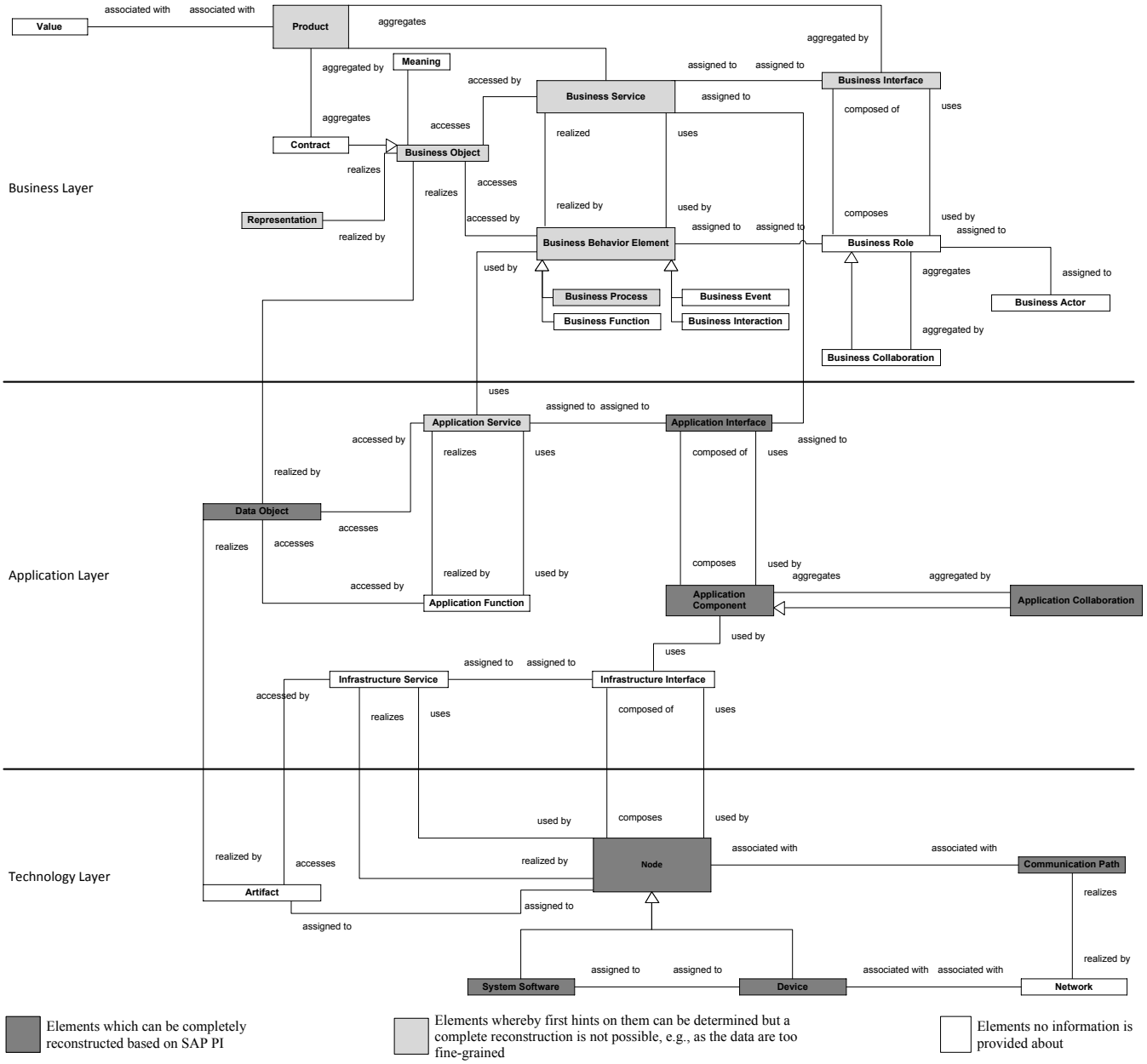

Figure 6. ArchiMate information model elements covered by SAP PI

In contrast to the application layer, attention is only paid to external visible functionality (*infrastructure services*), which similar to the interfaces, are not stored in SAP PI. As a special feature of ArchiMate, *system software* ("*software environment for specific types of components and objects"* (The Open Group 2012) belongs to the behavioral concepts. In SAP PI, a

subset of installed system software is registered at the System Landscape Directory, including the following elements: *operating systems*, *database systems* and *technical systems*.

Finally, *artifact*, the sole informational element, represents "*a physical piece of information*" (The Open Group 2012). With the exception of files imported by the SAP PI components such as WSDL files for specifying interfaces, information about existing *artifacts* especially the *artifacts* the application components are realized by is not available.

Cross-cutting Aspects

Aimed at providing a better alignment with TOGAF, ArchiMate, in its newest version, provides two extensions including motivation aspects, e.g., stakeholders, drivers and changes as well as implementation and migration extensions ranging from portfolio management to gap analysis (The Open Group 2012). However, such organizational background information is not directly stored in SAP PI.

Summary of ArchiMate model coverage

The detailed comparison of SAP PI with the EA information demand of ArchiMate reveals differences in the support for the layers. Primarily, structural application and technology aspects can be determined which is mainly attributed to the operative system characteristics of SAP PI. The information model elements of ArchiMate covered by SAP PI are shown in Figure 6. Nevertheless, some behavioral information is implicitly contained in the structural elements, ranging from the description of operations in interfaces to naming conventions. While SAP PI provides indications on business objects, behavioral and hierarchical information are missing to a large extent. For some business related aspects first technical hints could help to identify and accelerate subsequent manual data collection steps.

Evaluation of other information models

Apart from ArchiMate an evaluation of CySeMoL and planningIT is performed. In contrast to ArchiMate composed out of aggregated information within and across the enterprise layers, CySeMoL is focused on a specific area – IT security (Sommestad et al. 2010). Thereby, business aspects are neglected to a large extent and only considered indirectly in one attribute *expected loss* representing the expected loss e.g., in the case of an attack. The modeling language is more focused towards application and technology information making the strong technical view of SAP PI of slight disadvantage. However, gaps exist in the area of possible attacks and existing countermeasures which can only be partly gained.

PlanningIT is an integrated software suite aimed at supporting different tasks of Business IT management provided by alfabet (Alfabet AG 2012). Compared to planningIT the scope of ArchiMate as well as of CySeMoL is lower, which is mainly due to the fact that both frameworks merely outline a starting point and require concretization. While ArchiMate defines the entities on a very general level, CySeMoL proposes only an abstract information model expecting the definition of a concrete one. The information model has a similar structure to ArchiMate suffering from similar problems: While application and technology architecture are well covered business information is abstracted to a large extent.

To sum up, SAP PI covers the EA information demand only partly with focus on application and technology information making additional sources necessary to completely meet the EA information demand.

FORMAL TRANSFORMATION OF SAP PI TO EA INFORMATION MODELS

Subsequently, we completed the theoretical comparison of SAP PI and selected EA information models by a prototypical implementation. To allow an automated creation of an EA information model instance and to achieve consistency between both models, a model transformation approach is pursued. For this purpose, all involved models need to be formally described, leading to the use of Ecore (Eclipse 2012b), the de-facto modeling standard within the Eclipse community.

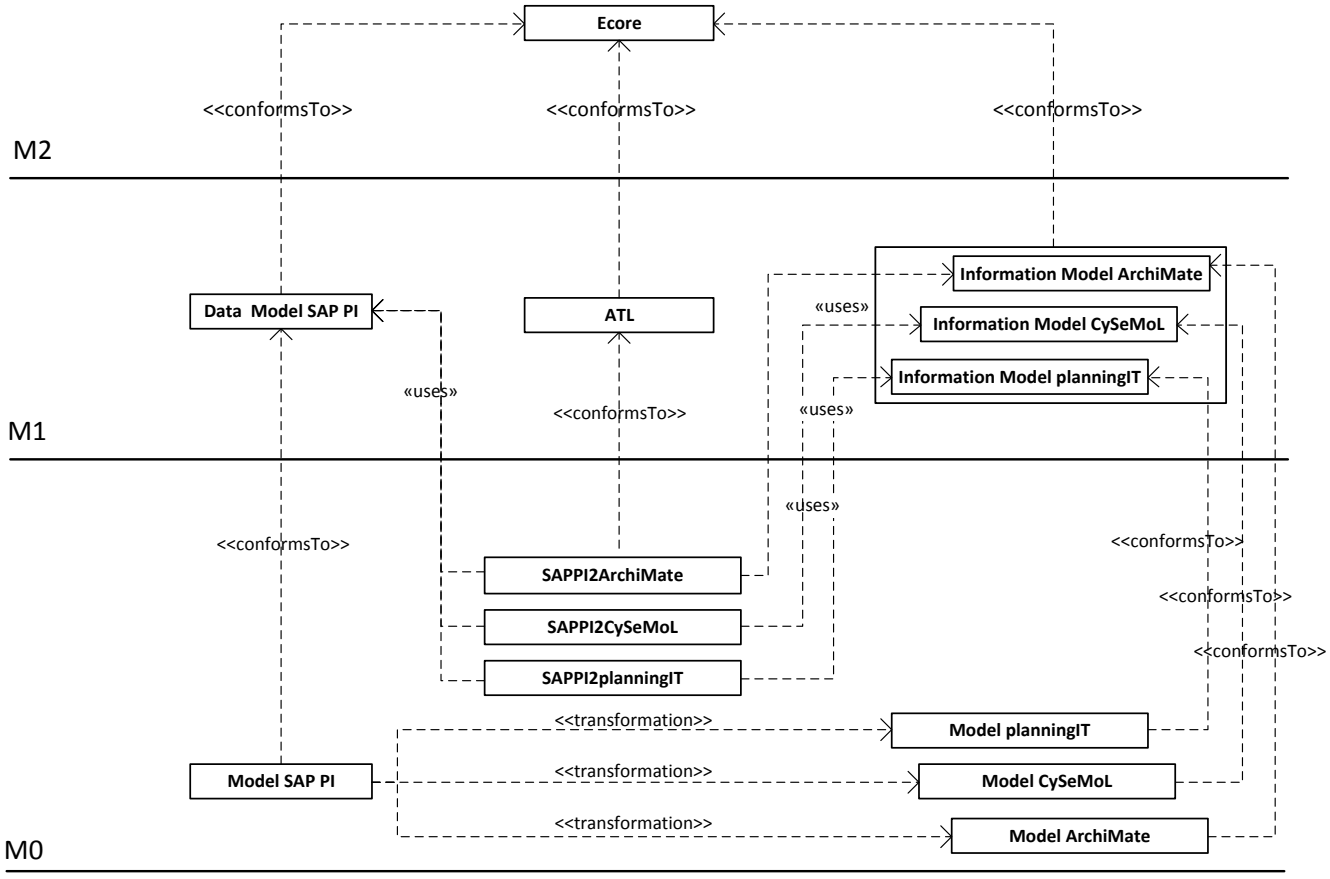

Figure 7. Different abstraction layers of the SAP PI data model and EA information model transformations

For selecting a transformation language several requirements should be taken into account, including appropriate tool support, expressiveness, and an appropriate documentation. An analysis of various transformation technologies revealed that ATL (Eclipse 2012a) meets the requirements best. Deficits exist only regarding the sparse documentation which is compensated by the enormous amount of code samples.

Within the implementation, the previous defined, informal transformation rules expressed in natural language are formally specified in ATL to be processed automatically. Figure 7 provides an overview of the ATL transformation enabling to transform the SAP PI data content (left-hand side) to EA information models (right-hand side). We implemented the transformation rules for three information models: ArchiMate, CySeMoL, and planningIT. Each rule set requires an Ecore instance describing the corresponding information model and an ATL script.

The initial target was a complete declarative solution deviated in some cases due to complex processes making imperative constructs necessary. Figure 8 exemplifies model transformations (dashed lines) on the basis of practical data provided by a German fashion enterprise. Visible is the business system *MOL* converted into one application collaboration on the ArchiMate side. The corresponding sub-software products *MI* and *PI* are mapped onto application components. These allow determining the subcomponents of the Application Collaboration *MOL*. The last example transformation relates the provided inbound interface to an application interface offered to the environment by application component *MI*.

As an example, Figure 8 further illustrates a simplified, declarative ATL matched rule generating an Application Component (to) out of an *Installed Software Product* (from). The rule is composed of a source pattern specifying the source object (*Installed Software Product*) in the SAP PI information model which is transformed into a target object (*Application Component*) whereby the name is taken over.

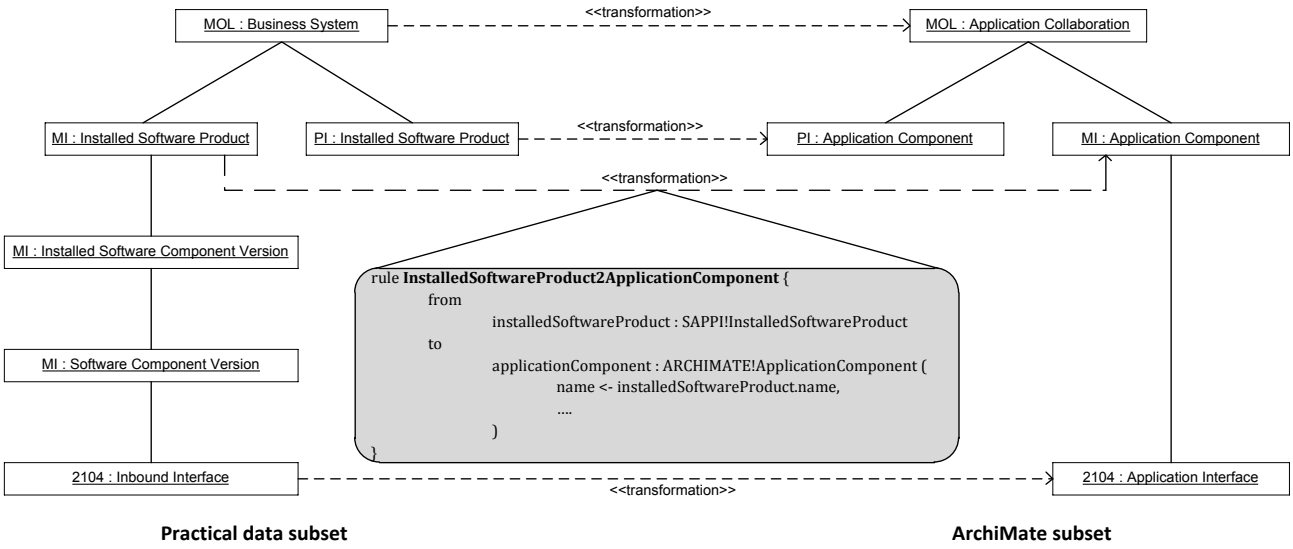

Figure 8. Transformation example with an ATL rule

CONCLUSION AND OUTLOOK

In this paper we showed EA documentation highly benefits from an automated collection of existing information using a productive system as an information source. Figure 9 illustrates the model coverage of the EA information models in our approach. The majority of EA relevant information in an SAP PI system is located on the *infrastructure & database* layer and the application & information layer. Deviances in the degree of model coverage between the EA information models can be explained by their different purpose and nature. While CySeMol focuses on infrastructure aspects, ArchiMate in turn seeks to capture a holistic view on enterprise aspects. Except for the *organization & processes* layer planningIT had the lowest model coverage.

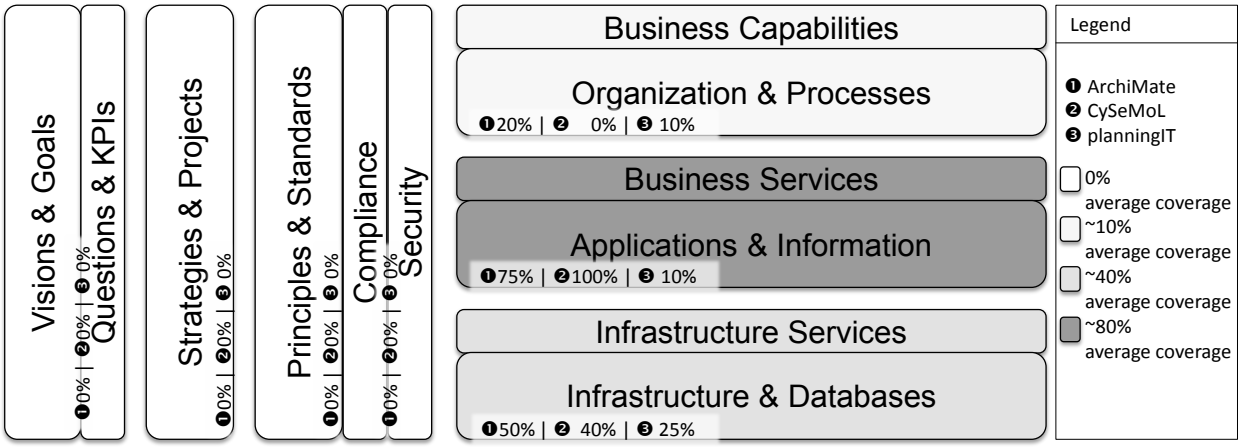

Figure 9. Model coverage of EA information automatically extracted from SAP PI

The extraction of EA information from a data source requires formal model transformations to automate documentation. This model transformation requires information models from both, source and target. In our example, we reverse-engineered an information model of SAP PI as a prominent example of an ESB system. The information is extracted from productive systems from a leading enterprise of the fashion industry and is automatically mapped with a transformation language.

The retrieved SAP PI information does not cover business aspects of all analyzed EA models. Additional data sources could provide *organization & processes* layer information to improve model coverage. Further research is necessary to reveal possible data sources in enterprises to extract EA information. Our research endeavor targets to automate large parts of the EA documentation. We expect to reduce documentation costs and improve data quality for decision makers.

REFERENCES

1. Alfabet AG (2012) alfabet. Available at: http://www.alfabet.com [accessed Feb. 28th, 2012].
2. Buckl, S. and Schweda, C.M. (2009) Future Research Topics in Enterprise Architecture Management - A Knowledge Management Perspective, *Trends in Enterprise Architecture Research*, Stockholm, Sweden.
3. Buckl, S., Matthes, F., Roth, S., Schulz, C., and Schweda, C.M. (2010) A conceptual framework for enterprise architecture design. *Trends in Enterprise Architecture Research*, Delft, Netherlands.
4. Buschle, M., Holm, H., Sommestad, T., Ekstedt, M., and Shahzad, K. (2011) A Tool for automatic Enterprise Architecture modeling, *Conference on Advanced Information Systems Engineering Forum*.
5. Department of Defense Architecture Framework Working Group & others (2007) DoD Architecture Framework, version 1.5. *Department of Defense*, USA.
6. Eclipse (2012a) Atlas Transformation Language. Available at: http://www.eclipse.org/atl/ [accessed February 29, 2012].
7. Eclipse (2012b) Eclipse Modeling Framework. Available at: http://www.eclipse.org/modeling/ [accessed Feb. 29th, 2012].
8. Farwick, M., Agreiter, B., Breu, R., Ryll, S., Voges, K., and Hanschke, I. (2011) Requirements for automated Enterprise Architecture Model Maintenance, *13th International Conference on Enterprise Information Systems*, Beijing, China.
9. Farwick, M., Agreiter, B., Ryll, S., Voges, K., Hanschke, I., and Breu, R. (2011) Automation Processes for Enterprise Architecture Management, *Trends in Enterprise Architecture Research*, Helsinki, Finland.
10. Fischer, R., Aier, S., and Winter, R. (2007) A Federated Approach to Enterprise Architecture Model Maintenance, *Enterprise Modelling and Information Systems Architectures – Concepts and Applications*, 2nd Int'l Workshop EMISA. Bonn, Germany.
11. Kaisler, S., Armour, F., and Valivullah, M. (2005) Enterprise architecting: Critical problems, *38th Hawaii International Conference on System Sciences*.
12. Kurpjuweit, S. and Winter, R. (2007) Viewpoint-based meta model engineering, *Enterprise Modeling and Information Systems Architectures-Concepts and Applications,* pp. 143–161.
13. Lee, Y.T. (1999) Information modeling: From design to implementation, *Second World Manufacturing Congress*, pp. 315–321.
14. Moser, C., Junginger, S., Brückmann, M., and Schöne, K.-M. (2009). Some Process Patterns for Enterprise Architecture Management, Strategy, pp. 19-30. Available at: http://subs.emis.de/LNI/Proceedings/Proceedings150/8.pdf [Accessed February 29th, 2012].
15. Nicolescu, V. (2009) Praxishandbuch SAP® NetWeaver PI – Entwicklung. SAP Press.
16. Rapid7 (2012) NeXpose. Available at: http://www.rapid7.com/products/NeXpose-enterprise-edition.jsp [Accessed February 29th, 2012].
17. Ross, J.W., Weill, P., and Robertson, D. (2006) Enterprise architecture as strategy: Creating a foundation for business execution, Harvard Business Press.
18. SAP (2009) SAP PI Best Practices: Naming Conventions. Available at: http://www.sdn.sap.com/irj/scn/go/portal/prtroot/docs/library/uuid/40a66d0e-fe5e-2c10-8a85-e418b59ab36a [Accessed February 29th, 2012].
19. Sommestad, T., Ekstedt, M., and Johnson, P. (2010) A probabilistic relational model for security risk analysis, *Computers & Security,* 29, p.659-679.
20. Spewak, S. and Hill, S.C. (1993) Enterprise Architecture Planning: Developing a Blueprint for Data, Applications, and Technology, Second Edition, John Wiley & Sons, London.
21. The Open Group (2009a) ArchiMate 1.0 Specification, Van Haren Publishing.
22. The Open Group (2009b). TOGAF Version 9 – A Manual, Van Haren Publishing.
23. The Open Group (2012). ArchiMate® 2.0, Available at: http://www.opengroup.org/archimate/ [Accessed February 28th, 2012].
24. Weill, P. and Ross J. W. (2009). IT Savvy – What Top Executives Must Know to Go from Pain to Gain, Harvard Business Press.

25. Wout, J., Waage, M., Hartman, H., Stahlecker, M., and Hofman, A. (2010) The Integrated Architecture Framework Explained: Why, What, How First. Springer, Berlin.
26. Zachman, J.A., 1987. A framework for information systems architecture. *IBM systems journal*, 26(3), p.276–292.

APPENDIX

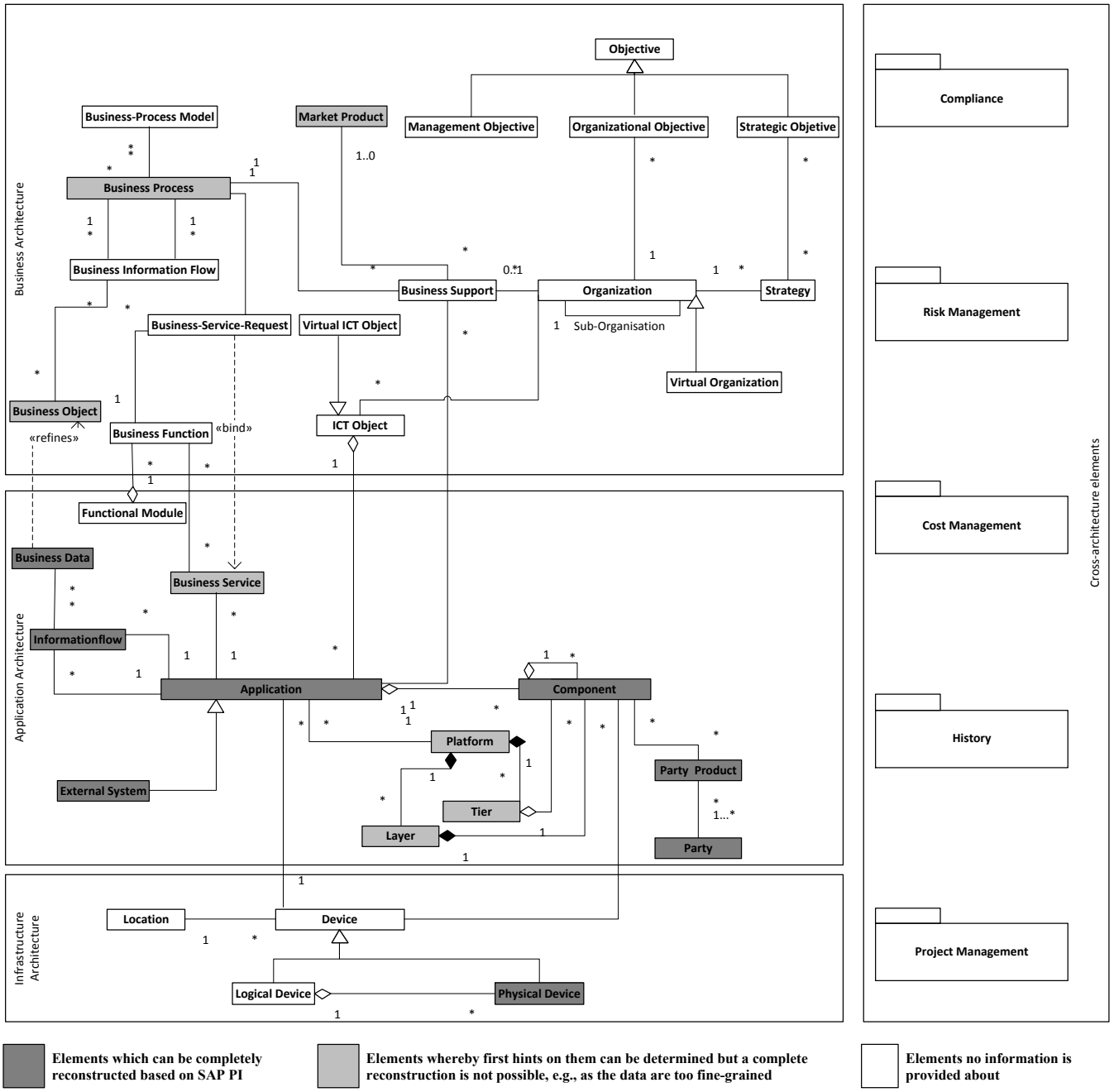

Figure 10. PlanningIT information model elements covered by SAP PI

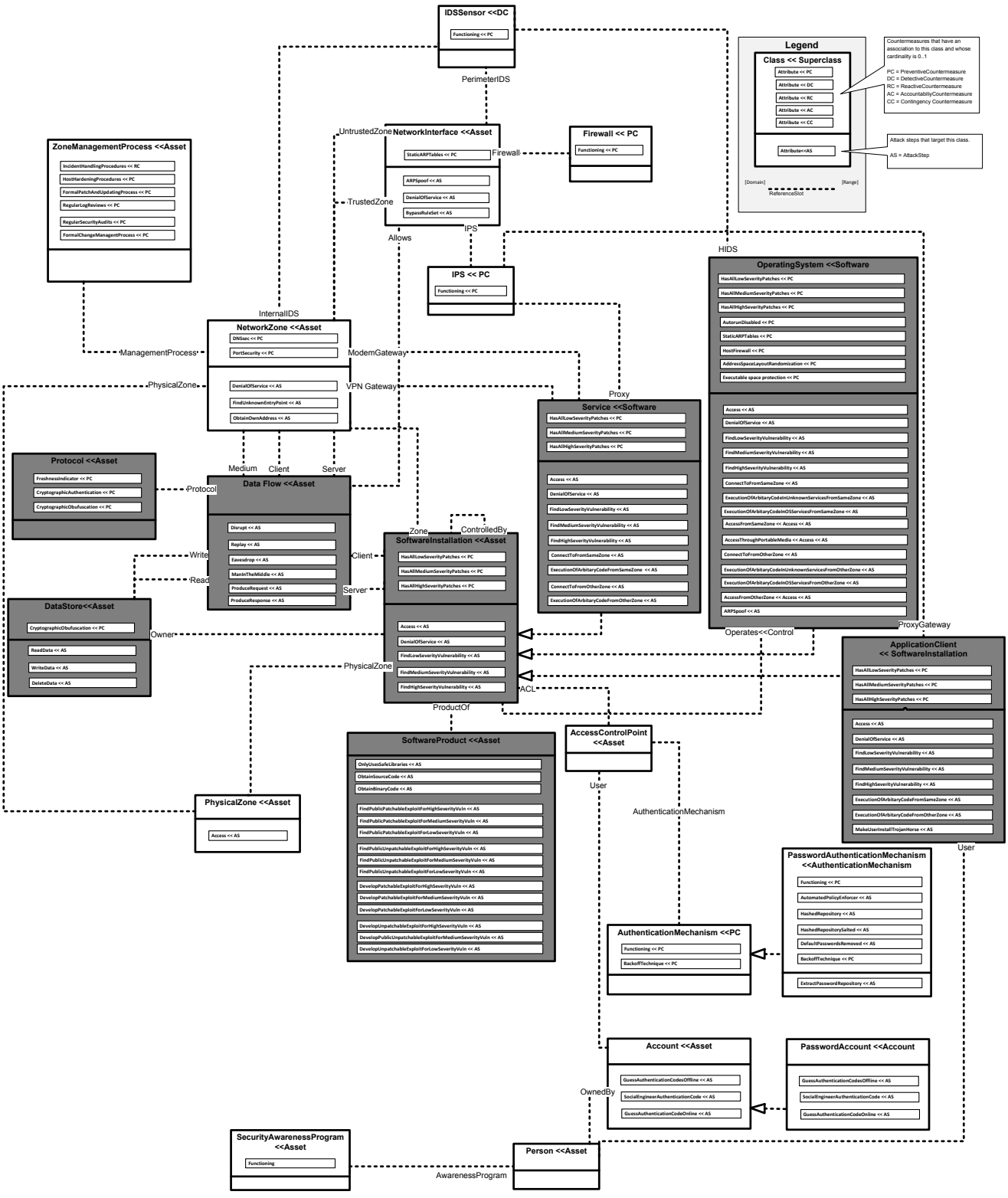

Figure 11. CySeMoL information model elements covered by SAP PI