Tool Support for Analyzing the Evolution of Enterprise Architecture Metrics

Manoj Bhat, Thomas Reschenhofer and Florian Matthes
Technische Universität München, Munich, Germany

Keywords: Enterprise Architecture Management, Metrics, Domain Specific Language.

Abstract: Managing the evolution of the Enterprise Architecture (EA) is a key challenge for modern enterprises. The EA metrics are instrumental in quantitatively measuring the progress of an enterprise towards its goals. Retrospective analysis of EA metrics empower business users to take informed decisions while planning and selecting efficient alternatives to achieve envisioned EA goals. Even though the current EA management tools support the definition and calculation of EA metrics, they do not capture the temporal aspects of EA metrics in their meta-model to enable retrospective analysis. In this paper, we first propose a model-based approach to capture the temporal aspects of EA metrics and then extend a domain specific language to compute EA metrics at any point of time in the past. This allows visualizing the evolution of EA metrics and as a consequence the evolution of the EA.

1 INTRODUCTION

Enterprises continuously change, improve, and evolve to respond to demands of their highly dynamic and competitive business and IT environment (Rouse, 2005). The enterprise architecture (EA) is an essential mechanism to capture requirements of their environment and it is widely accepted as a mechanism to achieve business and IT alignment (Matthes et al., 2008). Typically, an EA documents the (a) current state of the EA model, (b) target state as an envisioned long-term perspective, and (c) intermediate planned states to achieve the target state (Roth et al., 2013). By defining several planned states with respect to an envisioned target state, decision makers can define efficient alternatives for the desired change and use an EA metric model to justify their decisions. The EA metrics which are part of an EA model enable the measurement of the EA management (EAM) endeavor and support the quantitative based management of the EA targeting the predefined EAM goal achievement (Bose, 2006).

Managing the transformation of an enterprise from its current state to an envisioned target state via planned states is a challenge (Buckl et al., 2009). A key aspect in analyzing the managed evolution of an EA is the understanding of the evolution of its EA metrics which indicates the progress of an enterprise towards its target state. By considering EA metrics' evolution in a collaborative environment, decision makers can collectively reconfigure the planned states and select the appropriate alternative. Therefore, in this paper we focus on the temporal aspects of the EA metric model for analyzing the evolution of metrics and as a consequence the evolution of the EA.

In the context of an EA, a clear definition of the EA metric does not exist. In general, EA metrics aid in planning and controlling the structural and behavioral aspects of an EA. The EA metrics corresponding to the structural aspects include quality and acceptance-oriented metrics such as the project's quality plan availability (Matthes et al., 2012a). The EA metrics encompass Key Performance Indicators (KPIs) which correspond to the behavioral aspects of an EA (e.g. Application Criticality Rating). In this paper, the term metric refers to the EA metric that captures both the structural and behavioral aspects.

Even though metrics have been extensively used in Enterprise Performance Management systems for business intelligence, there exists a perception of the lack of metrics in EAM (Kaisler et al., 2005; Bose, 2006). This perception is changing as more and more efforts are being invested to formulate metrics and to map them with EAM goals (Matthes et al., 2012a). The survey conducted by Hauder et al. (Hauder et al., 2013), shows that the current EAM systems (85% of the surveyed tool vendors) support the definition, calculation, and visualization of EA metrics and 23% of

Copyright © 2015 SCITEPRESS (Science and Technology Publications, Lda.)

the surveyed systems allow the customization of metrics with a domain specific language (DSL).

Although the current EAM tools support the definition of metrics through DSLs, these DSLs however do not have the flexibility to compute the metric at a particular point of time in the past. Furthermore, majority of the existing EA tools maintain version repositories of EA models as a sequence of architectural snapshots (Buckl et al., 2009). One of the issues with the snapshot-based versioning is the incompleteness of the information stored in versioning systems to perform quality retrospective analysis. As versioning of EA models is not very frequent (quarterly, yearly, or only when significant changes are made), changes in the artifacts which affect business decisions could be lost during subsequent versioning of an EA. Moreover, the task of downloading and processing several versions from a repository is time-consuming and resource-intensive (Robbes and Lanza, 2007).

Therefore, a means to index appropriate business artifacts for quick access becomes vital in understanding the continuous evolution of an enterprise over a given period of time. The contribution of this paper is twofold: first we conceptualize the temporal aspects of metrics in an EA model. We then extend an existing DSL (Monahov et al., 2013) to facilitate the computation of metrics at any given time in the past by using the information model's history. In consequence, this allows the visualization of EA metrics' evolution for instance, on a time series graph.

Organizationally, Section 2 reviews the related work. Section 3 presents our approach to model the temporal aspects in an EA model. Section 4 presents a DSL to query the historized data and Section 5 presents the evaluation of our approach. In section 6, we conclude with a short summary.

2 RELATED WORK

Pourshahid et al. extend the Goal-oriented Requirement Language (GRL) with the concept KPI to measure business processes against business goals (Pourshahid et al., 2007). The meta-type KPI is modeled with attributes including *target value*, *threshold value*, and *worst value*. The concepts in the GRL model such as *goals*, *tasks*, and *actors* are associated with the KPI to provide the organizational context.

Popova and Sharpanskykh propose a meta-model to capture the goal, KPI structure and relations between them (Popova and Sharpanskykh, 2010). The meta-model captures the KPI as a concept with attributes including *name*, *definition*, *type*, *timeframe*, *scale*, and *threshold*. The attribute *type* captures the unit (continuous or discreet) used to measure the KPI. The attribute *timeframe* indicates the duration during which the KPI is defined and the attribute *scale* indicates the unit of measurement.

Strecker et al. present a method (MetricM) along with a DSL named MetricML (Strecker et al., 2012). The MetricM is integrated with the Multi-Perspective Enterprise Modeling method to enrich the description of the KPI with relevant enterprise context. The concept *indicator* in the MetricML meta-model refines the definition of the KPI as captured by Pourshahid et al. and Popova and Sharpanskykh.

To ensure consistency in defining, documenting, and retrieving KPIs, Matthes et al. introduced a uniform KPI description template to capture the *general structural elements* and *organization-specific structural elements* of a KPI (Matthes et al., 2012b). The general structural elements and organization-specific structural elements of a KPI are listed in Table 1 and Table 2 respectively. Furthermore, based on this template, an EAM KPI Catalog consisting of 52 literature-based and practice-proven KPIs was developed (Matthes et al., 2012a). This catalog also lists EAM goals and maps them to KPIs.

Table 1: General Structural Elements of a KPI.

Property	**Description**
Title	A unique name of the KPI.
Description	Detailed description of the KPI and its purpose.
Goals	Each KPI is related to at least one of the EAM goals.
Calculation	Textual description of how the KPI has to be calculated based on a certain information model.
Source	Source of the KPI (literature or practice).
Layers	A KPI can be assigned to one or more EA layers.

There also exists a large body of knowledge on DSLs for ensuring non-functional requirements (e.g. safety-critical concerns and quality assurance) of information systems through appropriate metrics. For instance, in (Monperrus et al., 2008) authors propose a language for calculating metrics that determine the internal complexity, size, and quality of the domain models. Similarly in (Klint et al., 2009), a DSL for analyzing the source code of software systems is presented. In the context of an EA, Iacob and Jonkers (Iacob and Jonkers, 2006) propose an approach based on the ArchiMate enterprise modeling language to quantitatively measure the performance of the EA models. Similarly, Johnson et al. (Johnson et al., 2007) extend

Table 2: Organization-specific Structural Elements.

Property	Description
Measurement frequency	The time interval between two measurement points.
Interpretation	Description of how the calculated value should be interpreted (good, acceptable, or bad).
KPI consumer	The person who is interested in the value of the KPI.
KPI owner	The person responsible for the KPI.
Target value	The KPI value to be achieved.
Planned values	The KPI values to be achieved while targeting the target value.
Tolerance values	The allowed deviations from planned and target values.
Escalation rule	Steps to be taken when the target EAM goal is not achieved.

the influence diagram notation (Expert, 2005) to support the quantitative analysis of the EA model properties including information security, performance, availability, and interoperability.

Monahov et al. (Monahov et al., 2013) propose a DSL named Model-based Expression Language (MxL) to formally define KPIs in the catalog (Matthes et al., 2012a) and to allow their automated calculation and evaluation. However, as discussed in (Monahov et al., 2013), the current version of MxL is not able to access the information model's history to compute the KPI at a point of time in the past.

Even though the aforementioned meta-models capture EA metrics and its properties, they do not consider the concept of time with respect to these metrics. Considering the temporal aspects in the meta-model is necessary for analyzing the evolution of metrics. Furthermore, it enables the extension of the existing DSLs to query the information model with respect to time and hence allows the calculation of metrics at any point of time in the past.

3 CAPTURING AN EA MODEL'S EVOLUTION

A wide variety of EAM tools provide the capability to model an EA and to analyze the data in a collaborative environment (Matthes et al., 2008). However, to address the challenges including the rigid information structures and mismatch between unstructured information sources in an EA, a model-based hybrid wiki approach (Matthes and Neubert, 2011) was designed and an EAM tool Tricia[1] was developed. We extend Trica in our approach for the following reasons:

1. Tricia follows a model-driven approach to system implementation and has a flexible meta-model (Büchner et al., 2010).
2. Tricia's MxL provides the capability to define and compute metrics.
3. Tricia's user-related services including *access control*, *search*, *versioning*, and *schema evolution* allows controlled access to the information model's history and enables the computation of EA metrics at any point of time in the past.
4. Tricia's extensive enterprise-level collaborative environment supports iterative development and management of quantitative models.

First, we will extend the EA metric meta-model with the temporal aspects. This is followed by a discussion on the extended version of the MxL that computes metrics at any point of time in the past by accessing the information model's history.

Figure 1, provides an excerpt of the EA metric meta-model with the focus on the metric and its temporal aspects. A detailed description of the Tricia meta-model is discussed in (Büchner et al., 2010). In the following sub-sections, we will discuss the concepts relevant in our context and illustrate how these concepts enable to capture the evolution of metrics.

Business Type, Business Attribute Definition, Business Entity, and Business Attribute. The Business Type and Business Attribute Definition allow the definition of the information model's schema. As shown in Figure 1, the Business Type contains multiple Business Attribute Definitions. A **user** (constrained by the **Role**) can create multiple Business Entities representing a specific Business Type. A Business Entity can contain multiple Business Attributes belonging to a Business Attribute Definition. These concepts in the meta-model allow users to create their own information model and its schema at runtime.

Metric, Metric Function, Measurement Rule (MR) and Metric's Value. The concept Metric comprises of the general structural elements and organization specific structural elements as discussed in the previous section (not shown in Figure 1). The temporal aspects of metrics are captured by its corresponding MR. The *frequency* information captured in the MR is used to trigger temporal events to recalculate metrics and the *timeSeriesType* information is used to visualize the evolution of metrics. The concept MR captures the following attributes:

[1]http://infoAsset.de

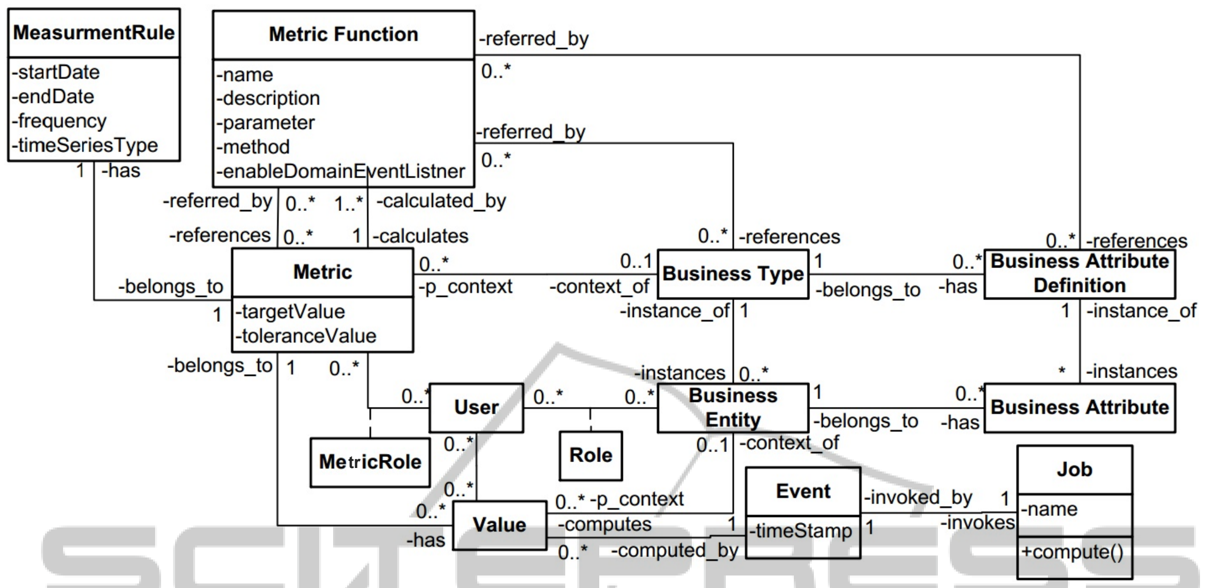


Figure 1: Meta-model capturing temporal aspects of EA metrics.

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS

- *startDate*: determines when to start monitoring a metric. The default value is the time when the metric was first defined.
- *endDate*: determines when to stop monitoring a metric.
- *frequency*: the interval between two measurement points; enumeration of values such as daily, monthly, and quarterly.
- *timeSeriesType*: enumeration of values such as continuous, discrete, linear, step-wise (Segev and Shoshani, 1987) for interpolation.

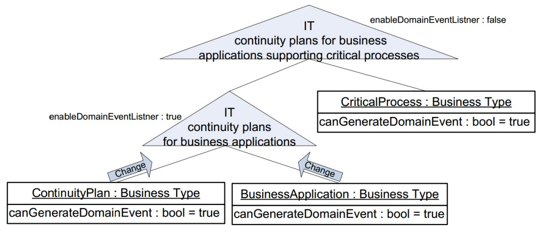
Furthermore, as shown in Figure 1, a Metric is calculated by a Metric Function which refers to multiple Business Types, Business Attribute Definitions, and other metrics for its calculation. The attributes of the Metric Function are as follows:

- *name*: name of the Metric Function.
- *description*: description of the Metric Function.
- *method*: implementation script in a DSL.
- *enableDomainEventListner*: if the value is true, the Metric Function listens to domain events. When a business entity triggers a domain event to re-compute a metric, the Metric Function is executed and the new metric **value** is persisted.

Events. We explicitly model events in the meta-model to enable the re-computation of metrics. On occurrence of a specific event, a new **job** is created by a job scheduler to recompute the corresponding metric. For better readability, dependencies from the source of an event to a specialized event are not shown in Figure 1. These events are classified as:

- *Temporal event*: Is triggered by the Measurement Rule depending on the measurement frequency of a metric, i.e. daily, monthly or quarterly.
- *User triggered event*: A user can trigger this event at any point to accesses the updated metric value.
- *Domain event*: A domain event is triggered by a Business Entity or Business Attribute when it is modified (created, updated, deleted).
- *Model change event*: Is triggered by a Business Type and Business Attribute Definition to represent a change in the information model's schema.

To avoid performance issues in the system due to frequent generation of domain events, a metric is recalculated only when the corresponding Business Type's attribute *canGenerateDomainEvent* and the Metric function's attribute *enableDomainEventListner* are set to true. In other words, domain events are propagated through the system and metrics are recalculated only when their corresponding flags are enabled. For instance, let us consider the scenario shown in Figure 2. The metric "IT continuity plan for business applications (ITC)" depends on the ContinuityPlan and BusinessApplication Business Types


IT continuity plans for business applications supporting critical processes
enableDomainEventListner : false
IT continuity plans for business applications
enableDomainEventListner : true
CriticalProcess : Business Type
canGenerateDomainEvent : bool = true
Change
Change
ContinuityPlan : Business Type
canGenerateDomainEvent : bool = true
BusinessApplication : Business Type
canGenerateDomainEvent : bool = true
Figure 2: Events.

whereas, the metric "IT continuity plan for business applications supporting critical processes (ITCP)" depends on the CriticalProcess Business Type and the value of the ITC metric. As both *enableDomainEventListner* and *canGenerateDomainEvent* flags corresponding to the ITC metric and its corresponding Business Types are set to true any change in instances of these Business Types results in the recalculation and persistence of the ITC metric. On the contrary, changes in instances of CriticalProcess or values of the ITC metric does not result in the recalculation of the ITCP metric as the corresponding *enableDomainEventListner* flag is set to false.

Role of User. Tricia provides access control and content authoring mechanisms to manage user permissions on the information model. The users or the group of users can *read*, *write*, or *administrate* content in the system. Similarly, user permissions and roles can be set on metrics defined in the system (through the **MetricRole** concept shown in Figure 1). The *metric consumers* have read access, while the *metric owners* are responsible for defining metrics and have both the read and write access.

Capturing the aforementioned concepts in the model allows persisting metrics as its corresponding business entities evolve and as a consequence allows the visualization of the metrics' evolution (Figure 8).

4 A DSL FOR ACCESSING AN EA MODEL'S HISTORY

By modeling the temporal aspects of metrics, we can query the value of metrics in the past and can visualize their evolution. However, if the temporal aspects of metrics are not yet modeled, their values in the past cannot be determined. In such scenarios, the information model's history needs to be accessed to compute the past metric values. Tricia manages versions of its entities and provides functionalities to compare versions and to restore previous versions of entities. As shown in Figure 3, versions are managed in changesets which not only capture the value of the entity before its change, but also its entity type, type of change (new, edit, delete), and modification timestamp.

An EAM tool that manages the information model's history and a DSL that can access the information model can facilitate evaluating a metric at any point of time in the past. As Tricia meets the above requirements, we extended the capabilities of the MxL to compute metrics at any point of time in the past. Furthermore, the MxL is independent of Tricia and can be integrated within other EAM tools. The MxL is specific to the EAM domain and is inspired

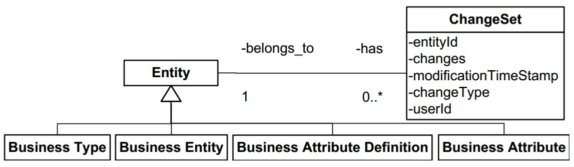


Figure 3: Versions of Entities.

by the Object Constraint Language (OCL) and Microsoft's Language Integrated Query (LINQ) leading to properties such as functional programming, object-orientation, and sequence-orientation. A detailed description of the available data types, language constructs, and operations in the MxL with examples is documented in (Monahov et al., 2013).

We extend the grammar of MxL by introducing a new literal "@" that retrieves the state of the object at a given time. That is, the MxL expression "*Expression @ time*" is not evaluated on the current state of the information model but on the information model's history and the state of entities at a given time.

The abstract syntax tree (AST) generated by the MxL parser containing the "@" literal sets the AtExpression as a node in the AST with one MxL expression as the left operator and the other MxL expression representing date as the right operator. This is illustrated in Figure 4 with a very simple example. The AST is passed to the MxL type checker to validate the *static semantics* of the language, which generates a Typed Expression Tree (TET) to be executed by the MxL execution engine. If the MxL expression refers to Business Types and Business Attribute Definitions,

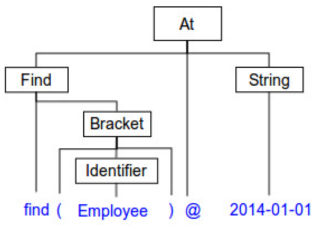


Figure 4: The AST containing the At Expression.

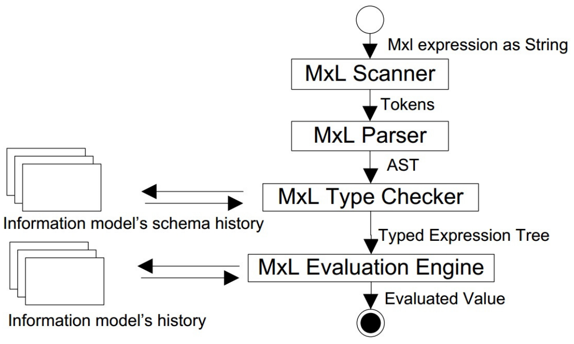


Figure 5: Evaluating MxL Expression with Date as String.

```
m1 = find(Employee).where(salary >3000).count()                    // 3
m2 = find(Employee).where(salary >3000).count() @pastDate1         // 1
m3 = find(Employee).where(salary >3000).count() @pastDate2         // 1
m4 = find(Employee).where(salary >3000).count() @
  [Today, pastDate1, pastDate2, pastDate3]                 // [3, 1, 1, 2]
```

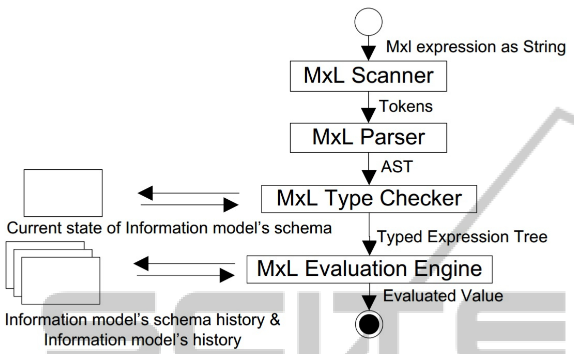


Figure 6: Evaluating MxL Expression with date as MxlObject/parameter.

the MxL type checker ensures their existence in the information model's schema and retrieves them before generating the TET. Please refer (Reschenhofer et al., 2014) for a detailed description on the MXL type checker and the type system of the MxL.

With the introduction of the AtExpression, the MxL type checker now has to determine the type of the MxL Objects on a given date. However, if the date itself is an MxL Object, the type of other MxL Objects corresponding to this date cannot be determined at compile time but only at runtime. Thus, we consider the following two scenarios:

Considering Date at Compile Time If the date is expressed as a string, the type checker ensures that the date is in a fixed format and determines the date at the compile time. As indicated in Figure 5, the type checker queries the schema history and checks the type of the objects corresponding to this date.

Considering Date at Runtime. As shown in Figure 6, if the date is an MxL Object, the type checker checks the type of all MxL Objects with respect to the current state of the information model's schema. At runtime, first the date is evaluated and then types of all other MxL objects are retrieved from the schema history and the expression is evaluated for this date.

Since Tricia maintains the change-sets of both the information model and its schema, it is possible to evaluate the expression irrespective of the change in either one of them. Let us consider an exemplary use-case to calculate a metric (*m* - *"find(Employee).where(salary>3000).count()"*). This metric computes the count of employees with

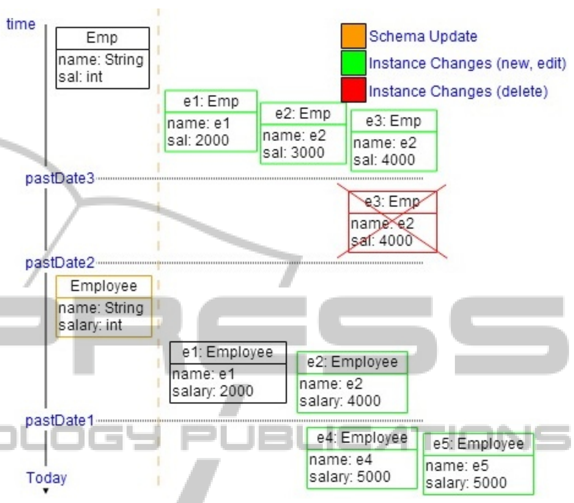


Figure 7: Evolution of Entity.

the salary more than 3000. Note that the value of the metric *m* changes as the business type *Employee* and its business attribute definitions *name* and *salary* evolve over time. As shown in Figure 7, evaluating the MxL expression (1) corresponding to *m1* at the current point in time (Today), returns value: 3 which is based on the current state of system. However, executing the MxL expression (2) representing the metric *m2* at the current point in time returns value: 1, which is based on the state of the system at a point of time in the past i.e. pastDate1. Furthermore, Figure 7 also shows that the schema itself is updated at some point after pastDate2 and executing the MxL expression corresponding to the metric *m3* with the date pastDate2, first checks for changes in the change-set representing schema change and returns the result corresponding to the schema at pastDate2. Also, the "@" literal can be followed by a sequence of dates, which returns a sequence of results corresponding to each date in the sequence as captured in mertic *m4*.

5 EVALUATION

We conducted an analytical and observational evaluation (von Alan et al., 2004) of the implementation of the MxL's At Expression. One of our industry partners in the financial sectors provided the histor-

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

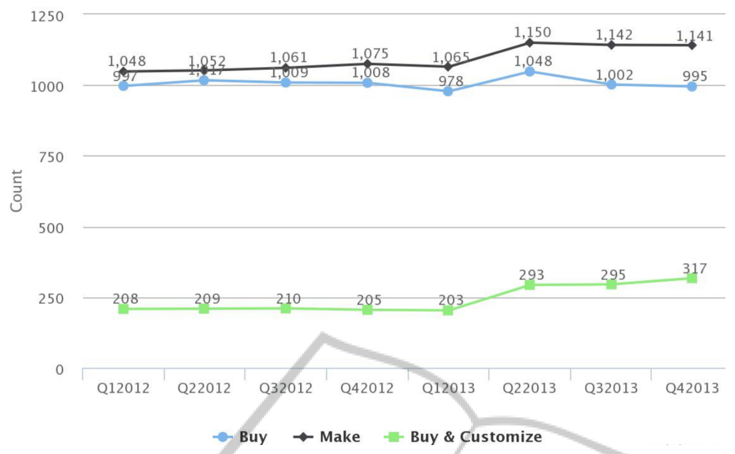


Figure 8: Evolution of NACL metrics.

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS

ized data for the evaluation. Without loosing generality, we imported the data with the timestamp of snapshots and updated the versions of artifacts to achieve a consistent view on the evolution of the information model. We then implemented metrics that are used to measure the complexity of the application landscapes (AL). These metrics (e.g., topology-based metrics and heterogeneity-focused metrics (Lagerstrom et al., 2014; Schuetz et al., 2013; Schneider et al., 2015)), were identified by our research group in a related research activity. The metrics used to measure the complexity of the AL include the *Number of Applications*, *Number of Information Flows*, and *Number of Applications per Customization Level (NACL)* metrics. The customization level indicates the category of the business applications such as *buy*, *make*, and *buy and customize*. The historized data received from our industry partner was versioned quarterly from the first quarter of 2012 to the fourth quarter of 2013. Using the "@" operator to compute the *Number of Applications* in the past, the trend in the metric is visualized in the Tricia platform. Similarly, the *Number of Information Flows* metric is visualized on a time series graph and is used to analyze the complexity and connectedness of the AL. The complexity of the AL is directly proportional to the number of interfaces (information flows) in each application. The *NACL* metric determines the number of business applications in different customization levels. The trend in the *NACL* metric, as shown in Figure 8, enables executives to compare and analyze where the company has been investing over the period of time.

6 SUMMARY, CONCLUSION AND OUTLOOK

In this paper, we have discussed the temporal aspects of metrics and proposed a meta-model to capture them in the context of the EAM. In section 3, we elaborated the concepts in the EA metric meta-model and discussed how metrics can be persisted as the information model changes. We then extended the existing DSL named MxL for calculating metrics based on the information model's history. In section 5, we presented the evaluation of our approach. Although we have successfully evaluated the EA metric meta-model and the extension of MxL in our research group, to comply to the observational evaluation as defined in (von Alan et al., 2004), we still need to study the usability and performance in a business environment. We propose to further collaborate with our industry partners from the banking domain for evaluating our approach with their application landscape data. Furthermore, conducting interviews of enterprise architects in the respective partner organizations will provide necessary inputs to improve our system before testing it in their practical settings.

In general, the time series analysis comprises of (a) developing the model that represents a time series and (b) predicting the future values based on that model. In this paper, our focus has been on the former aspect i.e. developing the model that represents the time series of metrics and analyzing their past evolution. However, considering both aspects of the time series analysis will further benefit decision makers in taking informed strategic decisions. The time series model of metrics will act as a base model to apply evolutionary algorithms to predict the future metric values along the time series. In the future work, we will validate this claim as well as evaluate the proposed EA metrics model in a practical settings.

REFERENCES

Bose, R. (2006). Understanding mgmt. data systems for enterprise performance mgmt. *Industrial Mgmt. & Data Systems*, 106(1):43–59.

Büchner, T., Matthes, F., and Neubert, C. (2010). Data model driven implementation of web cooperation systems with tricia. In *Objects and Databases*, pages 70–84. Springer.

Buckl, S., Ernst, A., Matthes, F., and Schweda, C. M. (2009). Visual roadmaps for managed enterprise architecture evolution. In *Software Eng., Artificial Intell., Networking and Parallel/Distributed Computing, 2009. SNPD'09. 10th ACIS Int. Conf. on*, pages 352–357. IEEE.

Expert, H. (2005). Hugin api reference manual, version 6.4, september 2005.

Hauder, M., Roth, S., Schulz, C., and Matthes, F. (2013). Current tool support for metrics in enterprise architecture management.

Iacob, M.-E. and Jonkers, H. (2006). Quantitative analysis of enterprise architectures. In *Interoperability of Enterprise Software and Applications*, pages 239–252. Springer.

Johnson, P., Lagerström, R., Närman, P., and Simonsson, M. (2007). Enterprise architecture analysis with extended influence diagrams. *Information Systems Frontiers*, 9(2-3):163–180.

Kaisler, S. H., Armour, F., and Valivullah, M. (2005). Enterprise architecting: Critical problems. In *System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii Int. Conf. on*, pages 224b–224b. IEEE.

Klint, P., van der Storm, T., and Vinju, J. (2009). Rascal: A domain specific language for source code analysis and manipulation. In *Source Code Analysis and Manipulation, 2009. SCAM'09. Ninth IEEE Int. Working Conf. on*, pages 168–177.

Lagerstrom, R., Baldwin, C., MacCormack, A., and Aier, S. (2014). Visualizing and measuring enterprise application architecture: An exploratory telecom case. In *System Sciences (HICSS), 2014 47th Hawaii Int. Conf. on*, pages 3847–3856.

Matthes, F., Buckl, S., Leitel, J., and Schweda, C. M. (2008). *Enterprise architecture management tool survey 2008.* Techn. Univ. München.

Matthes, F., Monahov, I., Schneider, A., and Schulz, C. (2012a). Eam kpi catalog v 1.0. *Technische Universität München, Germany, Tech. Rep.*

Matthes, F., Monahov, I., Schneider, A. W., and Schulz, C. (2012b). Towards a unified and configurable structure for ea management kpis. In *Trends in Enterprise Architecture Research and Practice-Driven Research on Enterprise Transformation*, pages 284–299. Springer.

Matthes, F. and Neubert, C. (2011). Wiki4eam: Using hybrid wikis for enterprise architecture management. In *Proceedings of the 7th Int. Symposium on Wikis and Open Collaboration*, pages 226–226. ACM.

Monahov, I., Reschenhofer, T., and Matthes, F. (2013). Design and prototypical implementation of a language empowering business users to define key performance indicators for enterprise architecture management. In *EDOCW, 2013 17th IEEE Int.*, pages 337–346. IEEE.

Monperrus, M., Jézéquel, J.-M., Champeau, J., and Hoeltzener, B. (2008). Measuring models. *Synthesis*, 19:20.

Popova, V. and Sharpanskykh, A. (2010). Modeling organizational performance indicators. *Information Systems*, 35(4):505–527.

Pourshahid, A., Amyot, D., Chen, P., Weiss, M., and Forster, A. J. (2007). Business process monitoring and alignment: An approach based on the user requirements notation and business intelligence tools. In *WER*, pages 80–91.

Reschenhofer, T., Monahov, I., and Matthes, F. (2014). Type-safety in ea model analysis. In *EDOCW, 2014 IEEE 18th Int.*, pages 87–94.

Robbes, R. and Lanza, M. (2007). A change-based approach to software evolution. *Electronic Notes in Theoretical Computer Science*, 166:93–109.

Roth, S., Hauder, M., Farwick, M., Breu, R., and Matthes, F. (2013). Ea documentation: Current practices and future directions.

Rouse, W. B. (2005). A theory of enterprise transformation. *Systems Engineering*, 8(4):279–295.

Schneider, A. W., Reschenhofer, T., Schütz, A., and Matthes, F. (2015). Empirical results for application landscape complexity. Technical report, Darmstadt Technical University, Department of Business Administration, Economics and Law, Institute for Business Studies (BWL).

Schuetz, A., Widjaja, T., and Kaiser, J. (2013). Complexity in enterprise architectures-conceptualization and introduction of a measure from a system theoretic perspective.

Segev, A. and Shoshani, A. (1987). Logical modeling of temporal data. In *ACM Sigmod Record*, volume 16, pages 454–466. ACM.

Strecker, S., Frank, U., Heise, D., and Kattenstroth, H. (2012). Metricm: a modeling method in support of the reflective design and use of performance measurement systems. *Information Systems and e-Business Management*, 10:241–276.

von Alan, R. H., March, S. T., Park, J., and Ram, S. (2004). Design science in information systems research. *MIS quarterly*, 28(1):75–105.