Use of a Domain Specific Modeling Language for Realizing Versatile Dashboards

Ulrich Frank
ulrich.frank@uni-due.de

David Heise
david.heise@uni-due.de

Heiko Kattenstroth
heiko.kattenstroth@uni-due.de

Chair of Information Systems and Enterprise Modeling
University of Duisburg-Essen
Universitaetsstr. 9, 45141 Essen, Germany

ABSTRACT

In order to make performance indicators a useful instrument to support managerial decision making, there is need to thoroughly analyse the business context indicators are used in as well as their mutual dependencies. For this purpose, it is recommended to design indicator systems that do not only include dedicated specifications of indicators, but that account for relevant relationships. In this paper, a DSML is proposed that enables the convenient design of consistent indicator systems at type level, which supports various kinds of analyses, and can serve as conceptual foundation for corresponding performance management systems, such as dashboard systems. Furthermore, indicator systems may also be used during run-time at the instance level to promote the distinguished interpretation of particular indicator values.

Keywords

Domain-Specific Modeling Language, Enterprise Modeling, Performance Management, KPI

1. MOTIVATION

In recent years, the increasing appreciation for performance indicators has promoted the idea of systems that provide users with performance related data. These systems, which we refer to as 'Performance Management Information Systems' (PMIS), are supposed to inform the individual user at a quick glance about the performance of entities such as an entire firm, specific business units, business processes, resources, and IT services. Inspired by technical metaphors such as 'cockpit' or 'dashboard', PMIS are more and more considered as a general instrument to foster managerial action, especially with respect to supporting, measuring, and monitoring decisions. The design of a PMIS implies the conception of indicators and systems of interrelated indicators ('indicator systems'). Indicator systems are usually defined by (top) management – with no regard of how they could be represented in an information system.

Current PMIS, such as dashboards, predominantly focus on the visualization of indicators that are considered to be relevant for certain decision scenarios. For this purpose, dashboard systems provide generic visualization 'gadgets', e.g., speedometers, traffic lights, or bar charts, that are usually applied to data originating from databases or files. However, to design PMIS that effectively support mangerial decision making, focusing on visualization only is not sufficient [3]. Instead, there is need to analyze what concepts are required to structure and effectively support a targeted decision.

Moreover, the design of indicator systems is not trivial. Already the specification of an indicator does not only require a profound understanding of the corresponding decision scenario and the relations to other indicators, but also recommends taking into account how an indicator affects managerial decision making [16, 19]; if managers regard an indicator as an end in itself, it will result in opportunistic actions that are likely not compliant with the objectives of a firm. This is even more important, because managers and other stakeholders are incited to predominantly align their behavior with specific (maybe mandatory) indicators and associated target values only [12, 17, 20]. If PMIS do not adequately address these challenges, they are likely to fail their purpose.

In this paper, we present an approach for PMIS that incorporates a domain-specific modeling language (DSML) for designing expressive and comprehensible indicator systems as core element. The DSML, called ScoreML, aims at promoting transparency, especially with regard to counter dysfunctional effects of indicators such as opportunistic behaviour. Also, indicator systems created with the ScoreML serve as a conceptual foundation for developing corresponding software. In addition to this use at build-time, our approach makes use of indicator systems at run-time as well, for example, as a front end to instance level performance data.

The approach is based on a comprehensive method for enterprise modeling and consists of the following components:

- a domain-specific modeling method comprising a language for modeling indicator systems – the ScoreML – and a corresponding process model that guides its application;
- a modeling environment implementing a ScoreML editor that is integrated with further editors for domain-specific modeling languages that are part of the enterprise modeling method;
- a software architecture for PMIS, in which the modeling environment constitutes the core component and that allows for integration with existing information systems.

Figure 1 illustrates the components of the PMIS. In this paper, we focus on the modeling language and its utilization in the context of the envisioned systems architecture. The other components are briefly discussed. The remainder is structured as follows: We derive domain-specific requirements for PMIS in Section 2. The prospects of our approach are illustrated in Section 3. The conceptual foundation, i.e.,

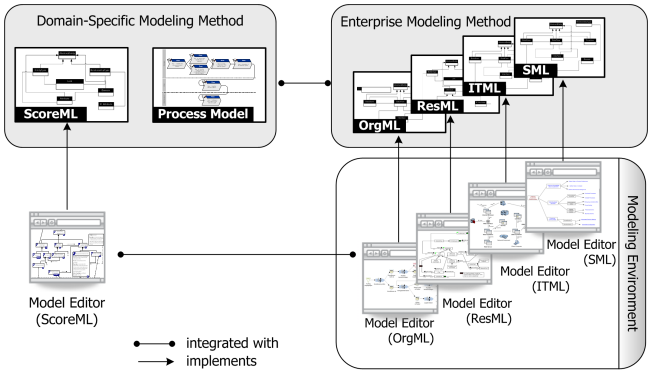

Figure 1: Components of the PMIS

meta-model and language architecture of the ScoreML, are presented in Section 4; the architecture for a model-based PMIS is envisioned in Section 5. Related work is discussed in Section 6. The paper closes with an evaluation, concluding remarks, and an outlook on future work in Section 7.

2. PMIS: REQUIREMENTS

An analysis of the current practice of dealing with indicators reveals a number of shortcomings. Based on these deficiencies, requirements for the domain-specific modeling language as well as for the envisioned architecture of a PMIS, which implements the DSML and integrates it with existing tools, can be derived.

First, currently there is hardly support for systematically creating and maintaining indicator systems available – indicators or indicator systems that are suggested in pertinent literature are usually described by informal concepts (e.g., [11, 15]). Hence, there is no linguistic support for guiding the construction of coherent indicator systems. This is a severe shortcoming: If an indicator system is partially inconsistent – e.g., includes incomplete indicator descriptions, undocumented dependencies, or even contradicting indicators – it jeopardizes its very purpose.

Req. 1 – Design of Indicator Systems: The design of consistent indicator systems should be promoted – if not enforced.

Second, the interpretation of indicators is crucial for well-founded decision-making. An adequate use of an indicator system implies a knowledgeable interpretation of the numbers that represent an indicator. Otherwise, a focus on 'meeting the numbers' (cf. [17]) may result in opportunistic behaviour, promote misleading conclusions and unfortunate decisions that – in the worst case – impede the overall enterprise performance.

Req. 2 – Business Context: To support the user with an appropriate interpretation of indicators, the indicator systems should be enriched with relevant context information to enhance the interpretation of indicators. This requires not only offering concepts that represent indicators, but also allowing for associating them with concepts that represent the business context (such as business processes).

Third, the utilization of indicator systems affects an enterprise and its employees at various – if not all – organizational levels, e.g., from executives at the strategic level to business units or IT experts at the operational level. The specific perspectives and levels of expertise vary among these groups of stakeholders. For instance, a process manager will have expectations that are different from those of an IT manager or an executive with respect to the types of indicators as well as the levels of detail and abstraction.

Req. 3 – Stakeholders: Meaningful presentations at different levels of abstraction are required to satisfy the needs of the multiple groups of prospective users. To foster an intuitive use of the language, concepts should be provided that these groups are familiar with.

Fourth, indicators used at different organizational levels are usually interrelated in that an indicator at a higher organizational level (e.g., strategic level) is often calculated from indicators at lower organizational levels (e.g., operational level). If not interrelated directly, indicator types can still be related indirectly, especially if the objects they meaasure are interrelated. Indicators, for instance, that measure the performance of business processes might be dependent on indicators measuring the performance of an information system underlying these processes, and thus are indirectly interrelated.

Req. 4 – Cross-Disciplinary Analyses: It should be possible to analyze interdependencies between indicators associated with different perspectives. This allows for making decisions on a more profound information base and for considering dependencies that go beyond the indicator system itself. Note that this request corresponds to the idea of the Balanced ScoreCard [12].

Fifth, supporting decisions requires particular indicator values, i.e., instance level data. There is a wide range of tools that aim at preparing and presenting these values, e.g., from dataware house to monitoring to reporting tools. However, they usually do not support users in interpretation and assessment of the presented values. Associating indicator values with the corresponding conceptual level – i.e., with the indicator system they are instantiated from and that are integrated with the relevant business context (cf. *Req. 2*) – contributes to a more sophisticated appreciation of indicator values.

Req. 5 – Instance Data: Tools for modeling indicator systems should be integrated with systems that manage corresponding instance level data (or integrate a corresponding component). It should be possible to navigate from the instance level to the conceptual level – and vice versa.

Sixth, PMIS usually visualize indicators in various ways. However, the cognitive styles of the involved users differ. Furthermore, different decision scenarios require different visualizations [2, 4]. In some cases, already the fact that an indicator is over (or below) a pre-defined threshold matters. In other cases, the focus is on performance over time, or the measured indicator needs to be compared to pre-defined thresholds.

Req. no.	*Description of Requirement*
Req. 1	Promote design of consistent indicator systems
Req. 2	Offer concepts for business context
Req. 3	Provide abstractions for different stakeholders
Req. 4	Enable cross-disciplinary analyses
Req. 5	Integrate type and instance level
Req. 6	Enable user-specific visualizations

Table 1: Summary of Requirements

Req. 6 – Visualization: Different stakeholders and different decision scenarios demand for versatile graphical representations of indicators. Therefore, it should be possible to adapt graphical visualisations to the individual needs of stakeholders without compromising the semantics of the represented concepts.

Table 1 summarizes the requirements for a performance management information system.

3. PROSPECTS OF THE APPROACH

The requirements pose the demand for an approach that supports the design and utilization of indicator systems in a systematic and structured manner. Conceptual models seem to be suitable, since they promise to reduce complexity by focusing on those aspects that are essential – and abstract from other. In this regard, the ScoreML promises more consistent indicator systems and allows for various analyses that – without such a support – a user can hardly perform. In the following, we illustrate the envisioned use of the DSML for designing and utilizing indicator systems at build-time as well as its potential for being leveraged as a 'dashboard' during run-time.

3.1 Focus on Build-Time

Users design indicator systems with the ScoreML by choosing indicators they consider relevant and adquate to support the targeted decision. At first, these indicators will be described on a more abstract level that is usually hardly quantifiable, e.g. "competitiveness". They can then refine these high-level indicators until they get down to a set of indicators that allow for expressive quantifications. By associating them to the objects they refer to, i.e., the reference objects they measure, the indicators are enriched with additional context information (cf. *Req. 2*). Once the indicator system is designed, (yet undiscovered) interdependencies among indicators can be elicited.

Figure 2 exemplifies this procedure for an IT Manager. It shows an indicator system model (top) and an excerpt of integrated IT resource/business process models (bottom). In the indicator system model, a few indicator types (attributes are omitted) for an IT-related indicator system are displayed. The indicator type *efficiency of IT department* is calculated from *IT costs* and *IT operations efficiency* (calculation rule is omitted, too). Some indicator types are associated to reference objects (an IT resource and two business process types). Here, different perspectives on an enterprise are accounted for: Two indicator types that are not directly interrelated (*IT costs* from an IT perspective and *average throughput time* from an operations perspective's indicator

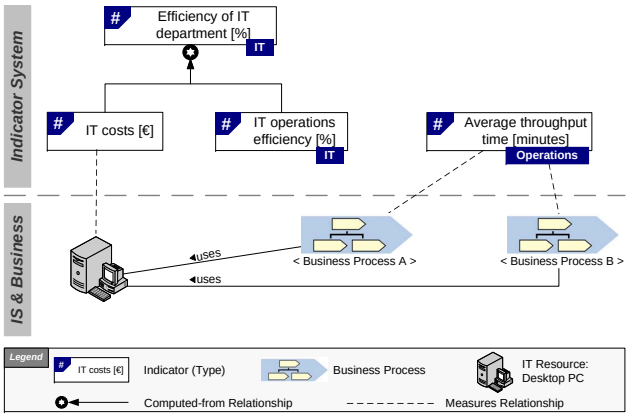

Figure 2: Short example including notation

system) are indirectly related; hence, an IT manager focusing on improving (e.g., reducing) *IT costs* only might, in the end, impede the performance of a related business process type's *throughput time*. If a timely execution of the process is more important than the IT costs of this resource, the additional business context information (cf. *Ref. 2*) and the capability to navigate through the other models (e.g., business process or IT resource models; cf. *Req. 4*) fosters decision-making (in this case of the IT manager). With regard to the ScoreML, the IT Manager could establish a specific association type between *IT costs* and *average throughput time* (e.g., 'influences') that visualizes the potential cause and effect relationship between these indicator types for further utilizations of the indicator system.[1]

3.2 Focus on Run-Time

Besides using models of indicator systems and the related models of the business context at build-time, they can also be used at run-time. A simplified example of an application scenario is illustrated in Figure 3.

A process owner, who is responsible for an online sales process, uses his personal dashboard to monitor the performance of the process. While the *daily revenue* corresponds to his expectations, the *average throughput time* (time between ordering and notification of the customer that the order has been approved) is exceeding its threshold. As a consequence, the currently running (active) instances of this process are affected, which are depicted in the lower section of Figure 3a.

To get a better understanding of the reasons for the dissatisfactory performance, he investigates ('drill-down') the indicators on which the critical indicator *average throughput time* depends, i.e., is calculated from. An example model of a business process *Online Sales* along with the required IT resources is displayed in Figure 3b.

The *process payment* activity is not functioning properly because the required part of the ERP-system is not available. The process owner escalates the problem to the IT staff, e.g.,

[1]Further, more extensive examples of indicator system models by means of ScoreML can be found in [7] and at http://openmodels.org/node/190.

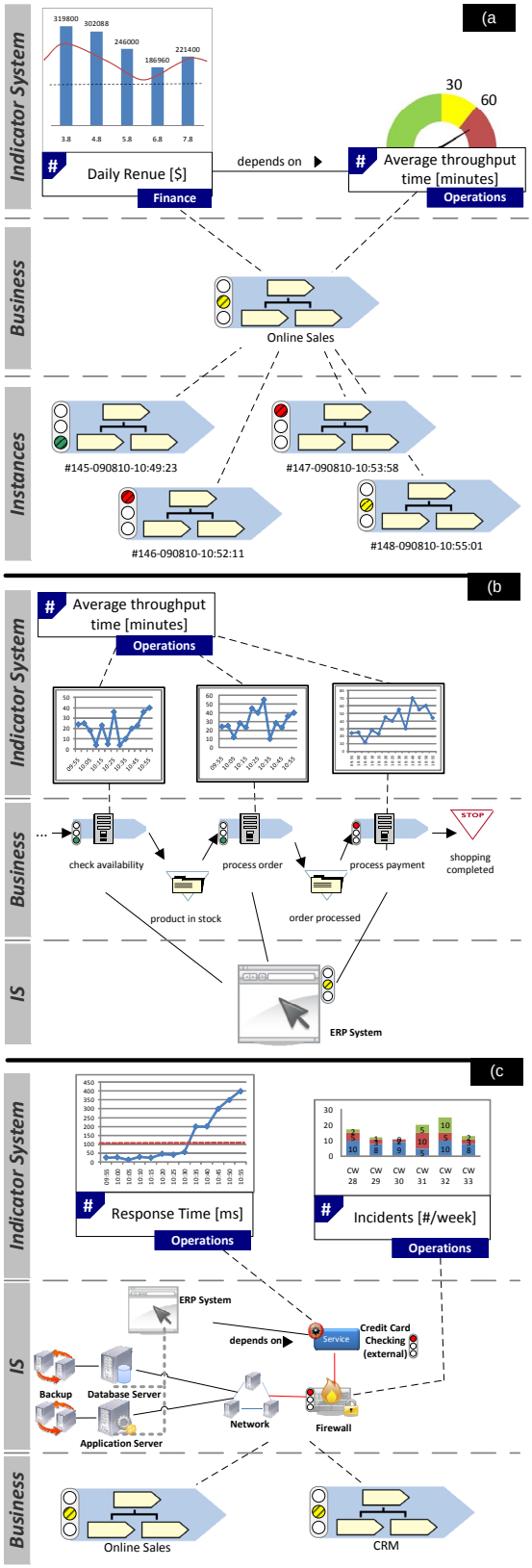

Figure 3: Dashboard for an Online Sales Process

through a ticket that refers to the particular business process and the malfunctioning IS. A member of the IT staff receives a notification about the incident. Hence, he uses his dashboard to assess the business impact (e.g., how many business processes are impacted? What is the loss of revenues to be expected in case of an outage?) of the incident. An excerpt of a model of the ERP system is displayed in Figure 3c along with corresponding indicators. The service *credit card checking* is offered by an external partner and securely accessed through a firewall. Obviously, the connection is not stable, i.e., has a high response time. Furthermore, the firewall was subject of several severe problems ('incidents') in the past weeks. Based on the information available, the user can decide what to do next, e.g., contact the vendor of the firewall and demand for a satisfactory solution.

4. MODEL-BASED PMIS

The ScoreML is based on a formal syntax and precise semantics, which provides two advantages over non-formal or general purpose approaches: First, it effectively supports and constrains users to build consistent and syntactically correct models (cf. *Req. 1*) as well as it facilitates convenient, intuitive, and secure modeling. Second, a DSML enables various kinds of analyses and transformations including code generation for corresponding software (cf.[13]). Furthermore, the ScoreML comprises a graphical notation with specialized, interchangable icons, which fosters communication between stakeholders with different professional backgrounds (cf.*Req. 3*).

4.1 Language Architecture

The approach we chose to develop the DSML is to enhance an existing method for enterprise modeling (EM) – the *multi-perspective enterprise modeling* (MEMO)-method [5] – by concepts and further components for designing and utilising indicator systems. MEMO consists of an extensible set of domain-specific modeling languages meant to model different aspects of an enterprise, such as corporate stratetgy (with the Strategy Modeling Language, SML; [8]), business processes (Organization Modeling Language, OrgML; [5]), resources (Resource Modeling Language, ResML; [10]), or IT resources (IT Modeling Language, ITML; [14]). MEMO is multi-perspective since it provides different views on various aspects of an enterprise. The MEMO languages are integrated in two ways (cf. [6]): First, they are integrated by a common meta meta model (M_3), so that the DSMLs are based on the same language specification (meta language). Second, they share common concepts at the meta level (M_2), which enables the integration of different perspectives (and models) addressed by each DSML (e.g., a meta concept 'business process' in OrgML and ITML).

Figure 4 illustrates the language architecture of MEMO and the interrelations between the DSMLs and the corresponding models at type level (M_1). Integrating the proposed DSML with the MEMO framework allows to benefit from a variety of existing modeling languages. Thereby, it is possible to associate indicator systems with models of the business context (cf.*Req. 2*) and representations different groups of stakeholders are familiar with (cf. *Req. 3*). Furthermore, the integration of the models provides a foundation to enable cross-disciplinary analyses that span various perspectives (cf. *Req. 4*).

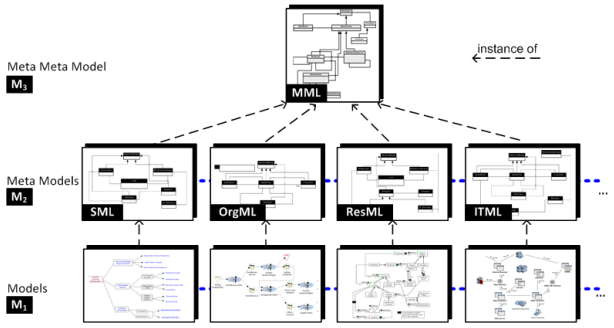

Figure 4: MEMO Language Architecture

4.2 Language Specification: ScoreML

The DSML is specified in a meta model using the Meta Modeling Language MML (cf. [6]). Figure 5 shows an excerpt of the ScoreML's meta model and depicts its main concepts. The specification of the language faced a number of challenges. Three important issues are addressed below.

First, ScoreML has to provide the users with a precise conception of indicators. From a modeling perspective, an indicator can be an attribute of an object, e.g., 'IT costs' of a piece of hardware; an aggregation of attributes of a collection of objects, e.g., the sum of 'IT costs' of all IT resources; or it can represent a snapshot of certain states over time, e.g., 'Average monthly IT costs'. In order to provide the user with convenient modeling concepts, we decided to introduce a specific abstraction rather than regarding indicators as attributes of objects or object collections. Such a conception includes core attributes and a differentiated conception of relationships between indicator types. This is realized by the meta type *Indicator* that comprises predefined attributes (e.g., name and description, purpose, potential bias) and a set of self-reflexive association types (e.g., computed from or similar to). We further introduced the association type *CustomizedRelationship* that enables users to qualify additional relations between indicators – for instance, an indicator can have an effect on another indicator (cf. Section 3.1). Additional customized attributes and non-'1..1'-associations can be realized by the meta types 'IndicAttributes' and 'IndicLink'.[2]

The second decision pertains to the flexibility and adaptability of indicators. The ScoreML has to allow users for adapting indicators to their individual needs and, furthermore, enrich indicators with additional semantics concerning the context they are used in. The former is addressed by distinguishing the concept indicator into the meta types *Indicator* and *SpecificIndicator*: While instances of *Indicator* represent generic information about an indicator type, *SpecificIndicator* allows users to assign this indicator type to specific reference object types, e.g., an indicator type 'average throughput time' assigned to 'business process type A' and to 'business process type B', including different values in the attributes benchmark, (avg.) value etc. In this context, the meta type *Threshold* allows for defining user-specific thresholds and corresponding notifications for a *SpecificIndicator*. This enables users to develop their individual 'performance dashboard' that includes indicators, thresholds, corresponding notifications and visualizations, and that fits to their personal cognitive style (cf. *Req. 6*). The latter – the need for additional semantics – is tackled by the meta types *ReferenceObject*, which is a surrogate for meta types like *BusinessProcess*, *Resource*, *Product* etc., and *DecisionScenario*, which enables the mapping of indicator types to scenario types (e.g., 'assessment of IT resources'). Note, the surrogate serves to illustrate the integration of ScoreML with the other MEMO languages: An indicator can be assigned to each (reasonable) meta type in one of the other DSMLs, which is the foundation for performing cross-disciplinary analyses. The re-use of concepts from other languages is denoted in the meta model with a rectangle at the concepts headers, including information about the origin (see the color legend in Figure 5).

Third, it is required to differentiate between types and instances of indicators: An indicator system contains types of indicators, while indicators that actually measure performance are instances. Especially with regard to *Req. 5*, it would not be satisfactory to neglect such instance level features. For example, a specific indicator type has a 'value' applying to a business process type (e.g., an average over all instances of this business process type); instances of this specific indicator type have a *particularValue*, describing the concrete value of a (projection of) process instance(s), e.g., at a certain time. To address this challenge, we make use of the concept 'intrinsic feature' [6]. An intrinsic feature – marked in the meta model with an 'i' printed white on black – is a type, an attribute or an association defined on meta level, but that reflects a characteristic we associate only to the instance level. Hence, although defined in the meta model this feature is not instantiated at type level but at instance level.

5. CORRESPONDING ARCHITECTURE

The outlined vision – designing and utilizing indicators in an versatile dashboard based on a DSML – requires an architecture for the PMIS that conforms to the requirements identified in Section 2.

First, there is need for a modeling environment that supports the user in designing and maintaining consistent indicator systems and, thus, implements the ScoreML. Figure 6 illustrates the modeling environment in the context of the PMIS architecture. It comprises a modeling editor for the ScoreML as well as – with regard to the integration with an enterprise modeling method – modeling editors for the other modeling languages.

Although the editors are separate, the underlying meta models are integrated (cf. Section 4). Thereby, the modeling environment maintains just a single model ('common model repository'), and the editors act on a defined set of concepts – i.e., parts – of this model. Hence, the 'surrogates' for reference objects in Fig. 5 are replaced by concrete meta types of other (MEMO) languages, like meta types for business processes or resources as indicated in the short example in Figure 2. This facilitates, e.g., cross-model integrity checks, since reference objects in the indicator system model refer-

[2]Note, the meta model at hand is a simplification due to the given restrictions of this paper. The full version can be found at http://openmodels.org/node/190.

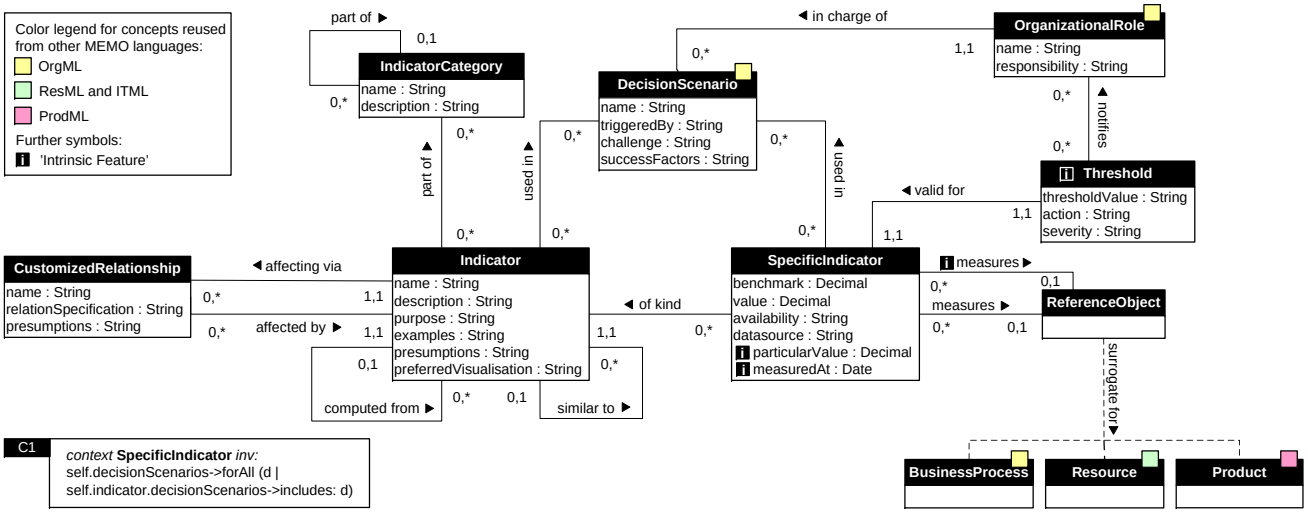

Figure 5: Excerpt of the ScoreML's meta model

ence to existing types in the model repository. Besides this model-based support during *build time*, the architecture also encourages using the models during *run time*. For instance, the language architecture of MEMO allows for navigating between different models (cf. *Req. 4*). Based on associations between concepts – i.e., instantiations of the associations between the corresponding meta types – it is possible to navigate from one model (e.g., an indicator system) to another (e.g., a resource model) by following the association between an indicator type and its reference object (in this case a resource type). The different modeling editors and the depicted language architecture are implemented in a modeling environment – called *MEMO Center NG*.[3]

Second, the tool requires a specific component for visualizing instance values of indicators (cf. *Req. 5*), e.g., by using the typical visualizations such as bar charts or traffic lights. In the architecture, it is represented by the 'dashboard' component. This component can be seen as visualization layer on top of the models (cf. [1, 4]). The proposed architecture poses one pivotal challenge that has to be addressed: The retrieval of instance values that are to be visualized on top of the models requires a connection to the information systems that manage the instance values.

On the one hand, the dashboard component can revert to historical data (e.g., for trend analysis) that are often stored in a data warehouse (DW). This data access requires to enrich an indicator type with a reference to the data source (e.g., tables in the DW) that contains its instance information. An example for a such a reference could be

```
'Select * from Database1.ITCosts
where DateTime between <BeginDate> and <EndDate>'
```

which retrieves the IT costs of a certain time period.

On the other hand, the instance data can be retrieved from operational information systems such as a Workflow Management Systems (WfMS), which contain information about process instances, an Enterprise Resource Planning (ERP) System, which holds information about the business objects such as orders, or a Configuration Management Database (CMDB) that is used to manage information of the IT resources in an enterprise.

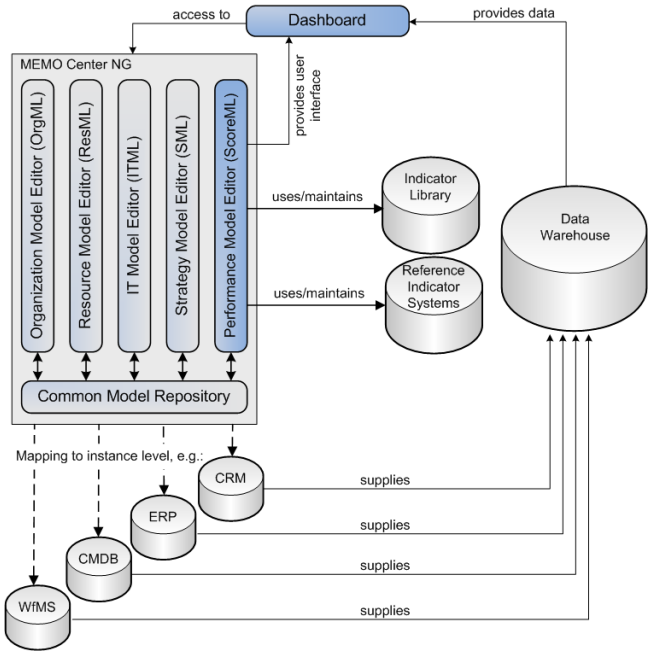

Figure 6: Proposed Software Architecture

The approach is complemented by an extensible set of *reference indicator systems* ('reference models') that are reconstructions of existing indicator systems (e.g., the 'DuPont' indicator system), and an *indicator library* that provides definitions of typical business performance indicators. Both can be loaded into the modeling environment as 'indicator

[3]More details on MEMO Center NG can be found in [6] and at http://www.wi-inf.uni-duisburg-essen.de/fgfrank/memocenter-en

systems building blocks', which serve as a basis for an enterprise specific adaption. More details on the purpose of reference indicator systems and the indicator library can be found in [7].

6. RELATED WORK

There are various information systems to support performance management. However, these tools often focus on the presentation of quantitative data, and they do not provide the user with additional information about indicators or their semantics (like their business context or effect-relations to other indicators; cf. *Req. 1 & 2*). In this regard, data warehouses store data that is extracted, transformed, and loaded from operational information systems to enable various analyses with respect to certain dimensions (like time, region, product) [9]. Thus, data warehouses provide a valuable data source for indicator instances (cf. Fig. 6). Unfortunately, data in data warehouses remain on a low level of semantics. Few approaches exist that try to augment data warehouses with additional context information. For instance, extensions of the EERM [21] or DSMLs [22] exist that allow for modeling dimensions for multi-dimensional structure of data and the navigation between these dimensions, e.g., roll-up or drill-down; hence, they complement our approach with respect to creating and maintaining the data warehouse underlying the PMIS architecture.

There are some commercial tools available (e.g., ARIS[4], ADOscore[5]) that also offer concepts for specifying indicators and – to some extent – allow for assigning them to concepts that represent the business context. However, their language specification is usually not available, and thus the concepts underlying the tool (i.e., the meta models) have to be reconstructed. As far as we can assess those tools, only ADOscore contains a more elaborate conception for indicators with respect to *Req. 1*. Unfortunately, this software is not (fully) integrated with the other ADO-modeling tools (esp., ADONIS), so it still lacks the integration of indicator systems with the business context.

When we developed the indicator modeling method in the context of MEMO, we built upon approaches that focus on indicator modeling (like [18, 23]) and extended those by (1) additional concepts for indicators (e.g., relations) and (2) concepts for the business context (cf. [7] for a more extensive description of these approaches).

7. EVALUATION & FUTURE WORK

In this paper, we outlined the domain-specific modeling language ScoreML for designing and utilizing indicator systems. The language is part of a PMIS that enables using the DSML not only at build-time, but also as versatile front end to instance-level performance data at run-time.

The PMIS consists of several components: a method for indicator modeling, which comprises the DSML and a corresponding process model, a modeling environment that implements the modeling languages, and a software architecture to enable the design and realization of versatile dashboards. In this paper, we focused on the modeling language and its utilization in the context of the envisioned systems architecture (the other parts are introduced in [7]). The design of the language and the corresponding architecture for PMIS were guided by six requirements:

The concepts of the ScoreML have been reconstructed from an existing technical language. Hence, the ScoreML provides its users with an elaborate linguistic structure that guides them with designing transparent and consistent indicator systems (*Req. 1*). By embedding the ScoreML into a method for enterprise modeling that also supports modelling of, e.g., processes, resources, and goals, the indicator models can be enriched with information about the relevant business context (*Req. 2*). Due to the integration of the ScoreML with other modelling languages (here: the family of MEMO languages), different perspectives on indicator systems are supported, e.g., from IT management and business management. This fosters collaboration and communication among the different stakeholders involved (*Req. 3*), as well as it facilitates the analysis of interdependencies between indicators associated with different perspectives (*Req. 4*). We further introduced means in language specification and architecture to integrate the indicator system models with tools that manage corresponding instance level data (*Req. 5*) and that enable the design and utilization of individual performance dashboards (*Req. 6*). Compared to the prevalent practice of creating indicator systems, the ScoreML is promising clear advantages. However, further studies are required to analyze factors such as acceptance and further conditions of successful use in practice.

In our future work we focus on further refining the language. This includes research on the technical integration between the modeling environment and the operational information systems. Also, we will enhance an existing library of reference indicator systems. We will also continue our work on model-driven development of versatile early-warning systems.

Acknowledgments

This research was partially funded by CA Inc. We thank our project partners for their support and valuable comments on preliminary versions of our research results.

8. REFERENCES

[1] S. Buckl, A. M. Ernst, J. Lankes, F. Matthes, C. M. Schweda, and A. Wittenburg. Generating Visualizations of Enterprise Architectures using Model Transformation. *Enterprise Modelling and Information Systems Architectures – An International Journal*, 2(2):3–13, 2007.

[2] W. W. Eckerson. *Performance Dashboards: Measuring, Monitoring, and Managing Your Business.* Wiley & Sons, Hoboken, NJ, 10 2005.

[3] S. Few. Dashboard Design: Taking a Metaphor Too Far. *DMReview. com. March*, 2005.

[4] S. Few. *Information Dashboard Design: The Effective Visual Communication of Data.* O'Reilly, Beijing, 2006.

[5] U. Frank. Multi-Perspective Enterprise Modeling (MEMO): Conceptual Framework and Modeling Languages. In *Proc. of the* 35^{th} *Hawaii International*

[4] http://www.ids-scheer.com
[5] http://www.boc-group.com

Conference on System Sciences (HICSS-35). Honolulu, 2002.

[6] U. Frank. The MEMO Meta Modelling Language (MML) and Language Architecture. ICB-Research Report 24, Institut für Informatik und Wirtschaftsinformatik (ICB), University of Duisburg-Essen, 2008.

[7] U. Frank, D. Heise, H. Kattenstroth, and H. Schauer. Designing and Utilising Business Indicator Systems within Enterprise Models – Outline of a Method. In P. Loos, M. Nüttgens, K. Turowski, and D. Werth, editors, *Modellierung betrieblicher Informationssysteme (MobIS 2008)*, volume 141 of *Lecture Notes in Informatics*, pages 89–106, 2008.

[8] U. Frank and C. Lange. E-MEMO: a method to support the development of customized electronic commerce systems. *Inf. Syst. E-Business Management*, 5(2):93–116, 2007.

[9] W. Inmon. *Building the data warehouse*. John Wiley & Sons, Inc. New York, NY, USA, 1996.

[10] J. Jung. *Entwurf einer Sprache für die Modellierung von Ressourcen im Kontext der Geschäftsprozessmodellierung*. Logos, Berlin, 2007.

[11] R. Kaplan and D. Norton. *Strategy Maps*. Harvard Business School Press, 2003.

[12] R. Kaplan and D. P. Norton. The Balanced Scorecard: Measures That Drive Performance. *Harvard Business Review*, 1992.

[13] S. Kelly and J.-P. Tolvanen. *Domain-Specific Modeling: Enabling Full Code Generation*. Wiley, New York, 2008.

[14] L. Kirchner. *Eine Methode zur Unterstützung des IT–Managements im Rahmen der Unternehmensmodellierung*. Logos, Berlin, 2008.

[15] R. L. Lynch and K. F. Cross. *Measure Up!: Yardsticks for Continuous Improvement*. Wiley, Blackwell, 1991.

[16] A. D. Neely, M. Gregory, and K. Platts. Performance measurement system design. a literature review and research agenda. *INT J OPER PROD MAN*, 15(4):80–126, 1995.

[17] B. Perrin. Effective Use and Misuse of Performance Measurement. *American Journal of Evaluation*, 19(3):367–379, 1998.

[18] A. Pourshahid, P. Chen, D. Amyot, M. Weiss, and A. Forster. Business Process Monitoring and Alignment: An Approach Based on the User Requirements Notation and Business Intelligence Tools. *10th Workshop of Requirement Engineering.*, pages 80–91, 2007.

[19] V. F. Ridgway. Dysfunctional Consequences of Performance Measurements. *Administrative Science Quarterly*, 1(2):240–247, 1956.

[20] J. M. Rosanas and M. Velilla. The Ethics of Management Control Systems: Developing Technical and Moral Values. *Journal of Business Ethics*, 57(1):83–96, 2005.

[21] C. Sapia, M. Blaschka, G. Höfling, and B. Dinter. Extending the E/R Model for the Multidimensional Paradigm. In *ER '98: Proceedings of the Workshops on Data Warehousing and Data Mining*, pages 105–116, London, UK, 1999. Springer-Verlag.

[22] Y. Teiken and S. Floering. A common meta-model for data analysis based on dsm. In J. Gray, J. Sprinkle, J.-P. Tolvanen, and M. Rossi, editors, *The 8th OOPSLA workshop on domainspecific modeling (DSM)*, 2008.

[23] B. Wetzstein, Z. Ma, and F. Leymann. Towards measuring key performance indicators of semantic business processes. In W. Abramowicz and D. Fense, editors, *BIS*, volume 7 of *LNBIP*, pages 227–238, Innsbruck, Austria, 2008. Springer.