Designing and Utilising Business Indicator Systems within Enterprise Models – Outline of a Method

Ulrich Frank, David Heise, Heiko Kattenstroth, Hanno Schauer
Chair of Information Systems and Enterprise Modelling
Institute for Computer Science und Business Information Systems (ICB)
University of Duisburg-Essen,
Universitätsstr. 9, 45141 Essen, Germany
{ulrich.frank|david.heise|heiko.kattenstroth|hanno.schauer@uni-duisburg.essen.de}

Abstract: The design of effective indicators and indicator systems requires a profound understanding of the relevant business context. Numerous relations and dependencies within an indicator system exist, which need to be analysed thoroughly: Many relations are based on implicit assumptions or sometimes not known by the management at all. This is of particular relevance for business success, since improperly used indicator systems may lead to 'dysfunctional effects' like opportunistic behaviour. This paper outlines a method for designing and utilising indicator systems. It fosters a convenient and consistent definition and interpretation of indicator systems. Furthermore, it serves as a conceptual foundation for related performance management systems, such as dashboard systems.

1. Motivation and Scope

In recent years, there has been an increasing demand for indicators to guide and justify management decisions. These business indicators – often referred to as Key Performance Indicators (KPI) – promise to reduce complexity and to promote a focus on relevant goals. Therefore, they are regarded by some as a key instrument of professional management, especially with regard to supporting, measuring, and monitoring decisions ([Si90], p. 12). For instance, in the field of Performance Measurement (PM) indicators are supposed to reflect aspects that are pivotal for certain decisions; therefore, strategies and goals are repeatedly refined and put in more concrete terms until they become measureable by KPIs. As a result, managers receive a set of various indicators that aim at the measurement of an enterprise's performance (called 'indicator system'). Well-known examples of indicator systems are *ZVEI* [ZE89] and the *RL* indicator system [Re06] for the financial domain as well as the *Performance Pyramid* [LC91] or the *Balanced Scorecard* [KN92] for the PM domain. Indicators are generally understood as quantitative measures that are specified using several mathematical constructs (e.g., ordinal or cardinal scale) for different types of reference objects and on various levels of abstractions in the enterprise (e.g., resources, processes, or business units; [Gr02] pp. 98ff.). By using indicator systems managers can define expected target values – e.g., by referring to benchmarks –, continuously monitor them and, in case of discrepancies, intervene into the enterprise's process execution.

However, the design of indicator systems is not a trivial task. Already the specification of an indicator does not only require a profound understanding of the corresponding decision scenario but also requires considering its relations to other indicators. Furthermore, it recommends taking into account how an indicator affects managerial decision making. If managers, for instance, regard an indicator as an end in itself, it will result in opportunistic actions that are likely to not be compliant with the objectives of a firm. This is even more important since indicator systems often have a strong influence on the enterprise [KN92], because managers and other stakeholders are incited to predominantly align their behaviour with specific (maybe mandatory) indicators and associated target values only. If indicator systems do not adequately address these challenges, they are likely to fail their purpose. Against this background, we assume that indicators should not be specified separately, but with particular attention to the relations that exist between them and to the business context they are utilised in. Both, complexity and criticality of the task to design systems of interrelated indicators recommend making use of a dedicated method that explicitly addresses those challenges. In this paper, we present a proposal of a method for designing and utilising indicator systems, which is based on a domain specific modelling language and aims at fostering transparency, validity, and reliability of indicator systems. It is intended to become part of the *multi-perspective enterprise modelling* method *MEMO* [Fr02] in order to facilitate semantically rich linking and integration of models of indicator systems with models of the corresponding business context – and vice versa.

The remainder of the paper is structured as follows: In section 2 important domain specific requirements for dealing with indicator systems are discussed. Based on these requirements, the elements of a method for the design and utilisation of indicator systems are presented in section 3, accompanied by first suggestions for a possible solution. The paper closes with related work (section 4) and an evaluation, concluding remarks, and an outlook to future work (section 5).

2. Business Indicator Systems: Requirements for a method

Based on assumptions about the quality of indicator systems used in business practise and resulting challenges, we develop requirements of the domain for the intended method.

The specification of an indicator system is a complex task that requires accounting for many aspects. Indicator systems that lack important aspects or are partially inconsistent jeopardize their very purpose.

> *Requirement 1*: The method should support – and if possible: enforce – the design of comprehensive and consistent indicator systems. For instance, there should be no contradictions or conflicts between indicators – if conflicts cannot be avoided, they should be made explicit (cf. [Ge86] pp. 114ff.).

The adequate use of an indicator system implies a knowledgeable and differentiated interpretation. For many decision scenarios a plethora of indicators is available

(cf. [LO06] p. 1). This hampers the selection of indicators that are significantly expressive regarding the underlying scenario, business strategies, and goals. Unsuitable indicators hold the dangers of promoting misleading conclusions and unfortunate decisions that – in the worst case – impede the achievement of the enterprise's goals (cf. [Ec05] pp. 197f.).

> *Requirement 2:* To support the user with an appropriate interpretation, the documentation of an indicator system should be enriched with relevant context information. This requires not only to offer concepts that represent indicators, but also to account for concepts modelling the business context (e.g., strategies, goals, business processes).

The adequacy of indicators as well as their relations (e.g., between themselves and to the goals) might be of question. Both are often based on subjective judgement and, thus, presumptive, whilst others are based on empirical evidence and 'notedly' objective (cf. [Gr02] p. 128; [LO06] p. 15; [Wa01] pp. 66f.; [So05] p. 226).

> *Requirement 3:* The method should provide concepts that allow for a differentiation of the rationale underlying an indicator and its relations.

Indicator systems are relevant for and used by various groups of stakeholders. The specific preferences as well as the level of expertise vary within these groups. For instance, a process manager will have different demands on indicators than an IT manager or a business manager regarding the types of indicators as well as the levels of detail and abstraction (cf. [Gl01] p. 8).

> *Requirement 4:* The method should support a meaningful representation of indicator systems on various levels of abstraction to satisfy the needs of multiple groups of prospective users. Further, the method should support the integration of an indicator system designed for a certain perspective with indicator systems of other perspectives.

For effective decision support, instances of indicator systems, i.e., concrete indicators should be provided and visualized by special tools such as dashboard systems.

> *Requirement 5:* The method should foster the construction of corresponding software systems by providing a conceptual foundation as well as guidelines for visualising indicators.

3. Elements of the method

The approach we chose to develop such a method is to enhance an existing method for enterprise modelling (EM) by concepts and further components for designing and utilising indicator systems. An enterprise modelling method in particular provides some prominent advantages:

- A (graphical) modelling language promises – similar to the use of conceptual models in software engineering – to support a rich and intuitive documentation, which fosters the communication between stakeholders with different professional backgrounds (cf. *Req. 4*). Further, it can depict manifold relations in a more comprehensible way than, e.g., a sequential textual description.

- A modelling language is based on a formal syntax and precise (if not formal) semantics, which allows for automated analyses, for transformations into models used for software development such as, e.g., object models for IT management software, as well as for interchanging indicator systems and their data between different information systems or enterprises (cf. *Req. 5*).

- By embedding it into an existing EM method, the proposed method additionally benefits from taking into account the relevant business context (cf. *Req. 2*).

Furthermore, we decided to use a *domain specific* modelling language, since general purpose modelling languages (GPML) like the 'Unified Modelling Language' (UML, [OM07]) or the 'Entity-Relationship-Models' (ERM, [Ch76]) show serious deficiencies with regard to our purpose (cf. [EJ01], [Lu04]). First, a GPML would not effectively support the construction of consistent models, since its syntax and semantics allows for expressing almost anything (*lack of integrity*). Second, it would be rather inconvenient to describe indicators using only generic concepts such as 'Class', 'Attribute', or 'Association' (lack of *convenience*) – the main concepts of the application domain are to be reconstructed by means of conceptual modelling in order to facilitate comfortable, intuitive, and secure modelling (cf. *Req. 1*). Third, the rather generic graphical notation (concrete syntax) of a GPML does not contribute to an illustrative visualisation (poor *comprehensibility of models*).

The method we suggest for multi-perspective indicator modelling consists of several parts. It is embedded in a context of additional abstractions and artefacts (Figure 1): The modelling concepts (abstract/concrete syntax) are defined in the Performance Modelling Language which we – within this paper – call SCORE-ML. A corresponding process model, which covers the entire lifecycle of indicator systems (*design*, *use* and *refinement*), guides the application of these concepts. In addition, there is an extensible set of reference indicator systems ('reference models') that are reconstructions of existing indicator systems (such as ZVEI) by means of the SCORE-ML. Further, the literature supplies a large variety of suggestions ('libraries') for indicators to be used in specific scenarios like – for IT management – the indicators suggested in the 'IT Infrastructure Library' (ITIL, [OGC07]). Both, the reconstructed reference systems and the method itself, may be linked to those indicator libraries to enhance descriptions of indicators and to indicate their relevance. On the other hand, indicator libraries may also make use of the method in order to provide a richer and more illustrative specification of indicators and to guide their appropriate use. Thereby, designers of indicator systems can take advantage of both reusing existing indicators or indicator systems, respectively, and of a domain specific modelling language that guides the construction or configuration of consistent models.

In this paper, we will focus on the method in the narrow sense (shaded rectangle in Figure 1): in section 3.1, we describe the main concepts of the SCORE-ML in their current state; in section 3.2, we introduce the process model; reference indicator systems and indicator libraries are briefly addressed in the last chapter again.

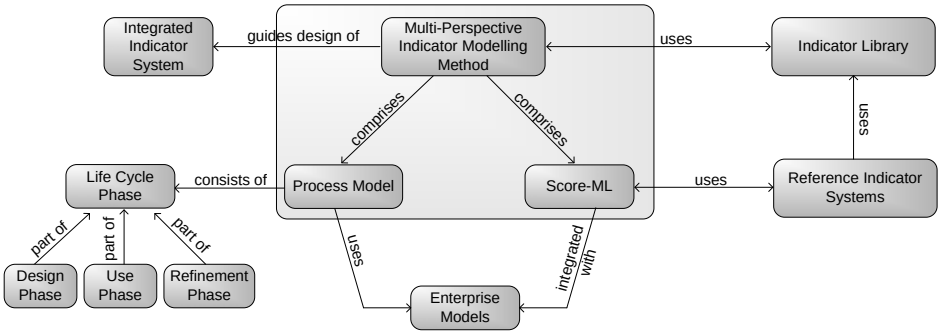

Figure 1 – Elements of the multi-perspective indicator modelling method

3.1 Modelling language: SCORE-ML

The specification of a language for modelling indicator systems faces a number of challenges.

Firstly, there is need to clarify the very conception of an indicator. From a modelling perspective, an indicator can be an attribute of an object, e.g., 'monthly revenues' of a product; it can be an aggregation of attributes of a collection of objects, e.g., the sum of 'monthly revenues' of all products; or it can represent a snapshot of certain features of an enterprise or the development of states over time, e.g., 'average monthly increase of revenues'. Since our focus is on the design of indicator systems, we decided to introduce a specific abstraction rather than regarding indicators as attributes of objects or object collections. This requires developing a conception of indicators that fosters a sophisticated specification. Such a conception includes core attributes and in particular a differentiated conception of relationships between indicator types. Secondly, we need to analyze how flexible and adaptable an indicator system is supposed to be. In the case of a database, the user can build individual indicators by using a data manipulation language (like in data warehouses). On the one hand, we do not want to leave the user alone on the level of a database schema, but provide him with more meaningful concepts to define indicators. On the other hand, it should be possible to adapt indicators to individual needs. Thirdly, it is required to differentiate between types and instances of indicators. An indicator system defines types of indicators, while indicators that are used to actually measure performance are instances. Fourthly, the semantics of indicators depends chiefly on the objects, they refer to, and the context they are used in. Hence, there is need for associating the concept indicator with reference objects and context information (cf. *Req. 2*). Figure 2 provides a suggestion for a meta model of the SCORE-ML that contains first – but not final – solutions for these challenges. Due to the given restriction in this paper we simplified the representation of certain aspects (see below).

Ad 1: To provide the users of the method with a rich, yet extensible concept of indicator, we defined a set of attributes and association types. Among others, indicator attributes include name, description, purpose, examples, availability (of required data), potential bias or preferred visualisation (e.g., 'traffic light', 'bar chart', 'speedometer'). Predefined association types for indicators comprise 'computed from' and 'similar to'. The relations realized by the meta type 'CustomizedRelationship' enable the qualification of further customized relations and, as a special case, cause-and-effect relations between indicators: The attribute 'relationSpecification' can be used to provide information about the type of the relation – for instance, an indicator can be a leading or lagging indicator with a positive or negative effect direction. The attribute 'presumptions' may be used – similar to the corresponding attribute of 'Indicator' – to provide information about the underlying rationale (e.g., 'empirical evidence' or 'subjective estimation' in combination with a statement about the prevalent assumptions). Note, this meta type as well as the association types are simplifications in tribute to the given restrictions of this paper. The original meta model of the SCORE-ML contains more differentiated meta types for such relations between indicator types, including a differentiation into bi-, unidirectional, and value driver relationships and a more differentiated arithmetic relationship (exemplarily denoted as 'computed from'). Additionally, a user can define individual attributes by instantiating the meta type 'IndicAttribute'; association types that are restricted in cardinalities other than the '1..1' in 'CustomizedRelationship' can be instantiated from the meta type 'IndicLink'. Note that these meta concepts are also not shown in the meta model excerpt.

Ad 2: An indicator system is supposed to represent all indicator types for a certain group of users. The system can be adapted to individual requirements on two levels. On the type level, an indicator type can be associated to a reference object type via the meta type 'SpecificIndicator'. This would allow one, for instance, to define an indicator type such as 'AverageCost' assigned to a business process type. On the instance level, an indicator can be defined by assigning it to a set of instances of a reference object type, e.g., to a projection on the instances of a business process type (see below). Reference object types can be further differentiated into 'dimensions' such as time period, regions etc. (cf. the multi-dimensional modelling in the field of data warehouses). Note that this is out of the scope of the indicator system itself. Instead, this kind of flexibility depends on the concepts applied to models of reference objects. On instance level, it depends on retrieval/navigation capabilities provided by corresponding information systems.

Ad 3: There are certain apparent features of indicators that we cannot express with the specification of indicator types, since they are used to represent instance states. Especially with respect to *Req. 5* it would not be satisfactory to neglect such instance level features. This applies, e.g., to the particular point in time an instance of 'SpecificIndicator' is created at or to its concrete values. To meet this challenge, we suggest the concept of 'intrinsic feature' [Fr08]. An intrinsic feature is an attribute or an association that reflects a characteristic that we associate with a type that applies, however, only to the instance level. Hence, an intrinsic feature within a meta model is not instantiated on the type level, but only one level further down, i.e., on the instance level. In the meta model in Figure 2, the intrinsic association 'measures' allows for assigning sets of instances of a reference object type to an instance of 'SpecificIndicator'. In the MEMO Meta Model-

ling Language [Fr08], which is used to specify the present meta model, intrinsic features are marked by an 'i' that is printed white on black.

Ad 4: To enrich indicator types with semantics, they can be associated to goals, decision scenarios, and reference objects. Every indicator type can be associated to a goal type. In an ideal case, there is an elaborate model of a company's goal system (which would be out of the scope of an indicator system). The meta type 'DecisionScenario' can be instantiated to a scenario type (e.g., 'assessment of IT resources'), which in turn can be instantiated to a particular decision. An indicator type ('SpecificIndicator') can also be applied to certain reference objects. In the meta model, this is indicated through the abstraction 'ReferenceObject', which serves as a surrogate for meta types such as 'BusinessProcess', 'Resource', 'Product' etc. Note that the attributes 'benchmark' and 'value' apply to the type level, e.g., a reference value and the actual value for the average throughput time of the business process type 'order management'. The intrinsic feature 'particularValue', on the other hand, serves to describe the value of a (projection of) process instance(s) – which may be interesting for a drill-down scenario in, e.g., a dashboard tool. The meta model includes a number of additional OCL constraints. The excerpt in Figure 2 shows only one constraint, which ensures that a 'SpecificIndicator' type can only be associated to a decision scenario type, if the corresponding 'Indicator' type is associated to the decision scenario type, too.

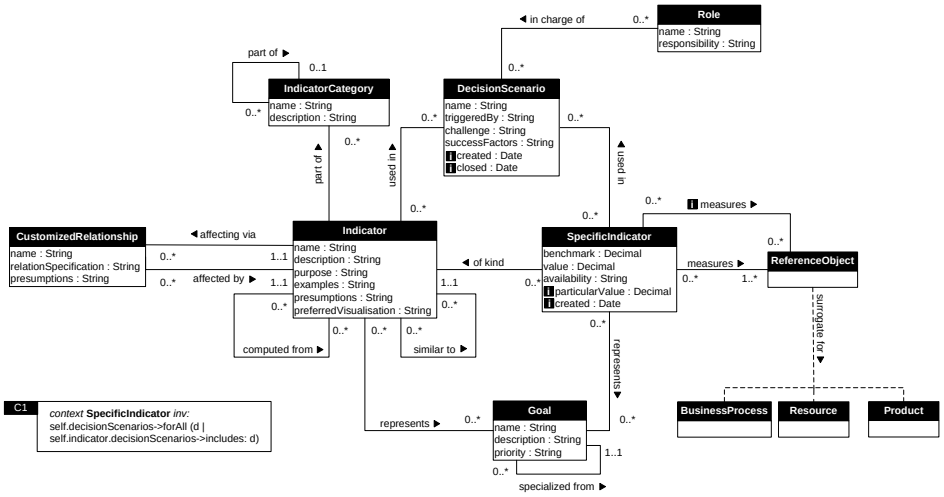

Figure 2 – Excerpt of the SCORE-ML meta model

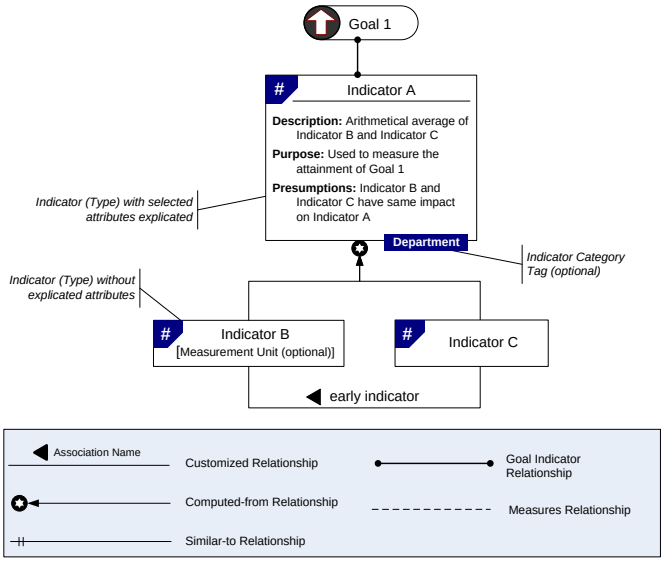

Figure 3 – Illustration of the graphical notation of the SCORE-ML

The specific notation of the SCORE-ML is illustrated in Figure 3. An exemplary model of an integrated indicator system is depicted in Figure 4. It shows the utilisation of the meta concepts goal (*Goal System*), indicators, and their relations (*Indicator System*). The business context is added in the '*IS & Business*' layer, showing the association of specific indicators to reference objects. We enriched one specific indicator and one customized relationship with a full description of their attributes in order to illustrate the structured documentation provided by the SCORE-ML. Further, the example includes an instance level representation (*Instances*): For business process *A*, two instances are shown. They are enriched with an additional symbol (traffic light-metaphor) that visualises the current status (i.e., instance status) of the assigned 'SpecificIndicator' (here: throughput time).

3.2 Corresponding Process Model

The application of the presented modelling concepts is embedded in a process model that envisions the use of the SCORE-ML either 'stand-alone' or in combination with existing enterprise models (e.g., resource, process, strategy, and other models).

The process model can be divided into three phases, according to the life cycle of indicator systems: The first phase is the *design* of the indicator system (conceptual), the second phase describes the *use* of the indicator system, and the third is the continuous *refinement* of the indicator systems (both: 'empirical'). Figure 5 gives an overview about the phases and their steps.

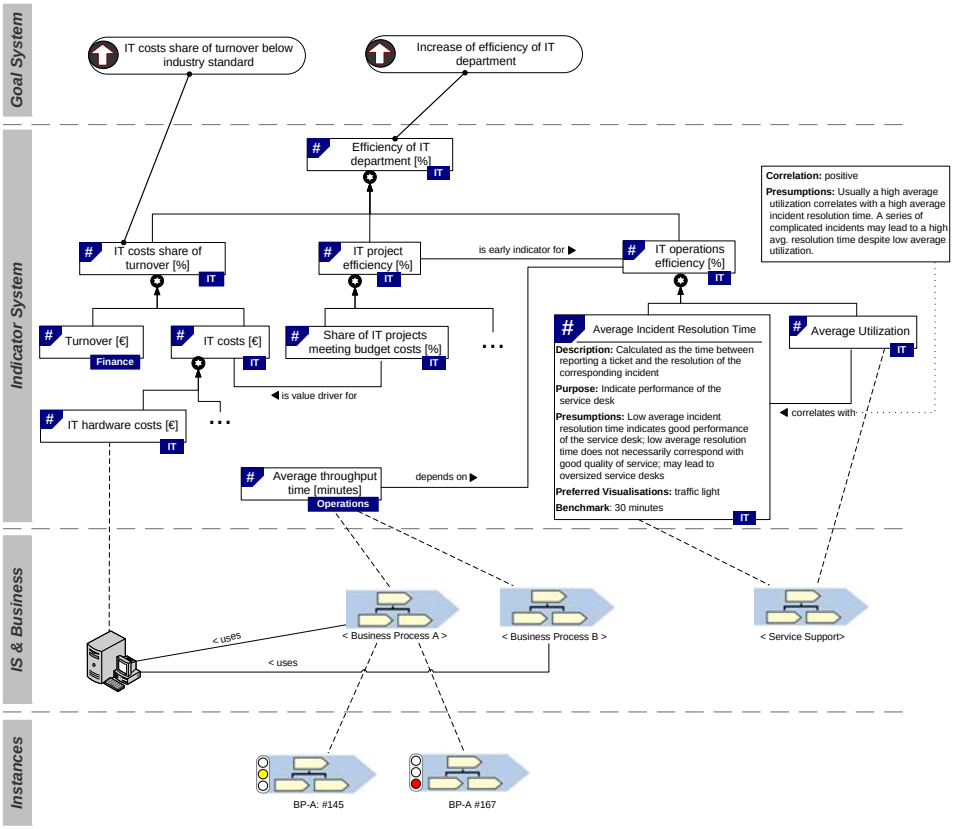

Figure 4 – Exemplary model of an integrated indicator system (inspired by [Kü06])

Design Phase

This phase can be differentiated into two shapes (cf. Figure 5): first, the 'real' design process (for advanced users, such as consultants and experts), wherein an indicator system is completely set up from scratch in order to match specific demands or realise advantage over existing indicator systems; second, a selection/configuration process for less ambitious users (e.g., end-users), wherein an existing indicator system – e.g., a reference indicator system – is reused and (where necessary) adapted for the specific needs. The latter one is mostly self-explanatory (and, for this reason, will not be further discussed in this paper). Hence, in the following we focus on the 'real' design process.

Step 1: Identification of relevant decision scenario, indicators, and data

In the process of setting up an indicator system, the identification of the relevant decision scenarios and corresponding indicators is one of the first and pivotal tasks. There are two directions to identify and define indicators with respect to a certain scenario: (i) defining indicators 'top-down' by operationalising strategies and goals ('What do we want to measure?'), and (ii) 'bottom-up' by finding measureable aspects ('What can we

measure?') or by using mandatory indicators that are provided by the management. Both directions are not necessarily disjunctive, they can rather be used together.

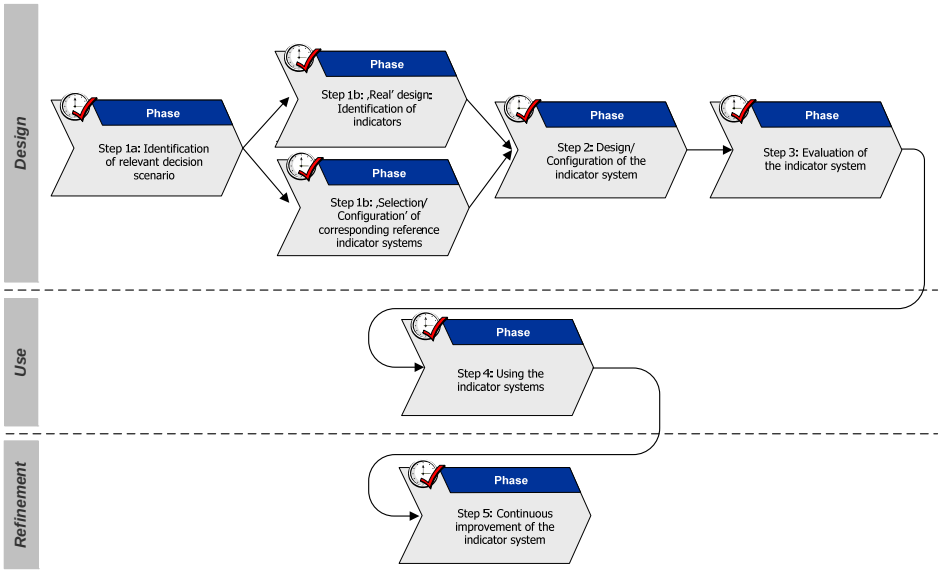

Figure 5 – Process model of the SCORE-ML

In the top-down direction, the enterprise's strategies and goals are refined until the matter of their measurement occurs. While some are apparently easy to measure (e.g., the increase in revenue), others are hardly measureable to a satisfactory extent. In SCORE-ML, each ambiguous indicator can (and should) be marked by means of the according attribute (*presumption*). Thereby, the user is able to express and, to a certain degree, forced to think about the potential distortions of the indicator. In the bottom-up approach, the user searches for all those indicators he *can* measure. In both approaches, the identification of indicators can be supported by enterprise models. For instance,

- (IT) resource models (such as provided by the MEMO RESML [Ju07] and the MEMO ITML [Ki08]) offer an overview about the resources, their main properties (e.g., capacity, costs) and their relationships; this also includes information about their usage in processes ('resource allocation').

- business process models (such as in MEMO ORGML, cf. [Fr02]) describe the course of action(s) in a company; they provide details about the activities like times (e.g., execution, transport), exceptions, costs, and risks.

- strategy models (such as in MEMO SML, cf. [FL07]) represent the company's strategies, goals, measures (in terms of strategic actions), and their relations, for instance, to the processes and resources in an enterprise (esp. supporting the top-down approach).

These models further support the search for potential indicators by showing which reference objects are of relevance: the business context provided by reference objects supports the user in assessing the relevance, usefulness, and appropriateness of a related indicator. Furthermore, the models also offer potential indicators in terms of attributes and quantifications (e.g., time-related attributes). Additionally, the aforementioned indicator libraries can serve as a basis for the identification of indicators: for certain domains and decision scenarios they offer lists of recommended indicators. Instead of simply applying these indicators directly, the user is supported in relating them to other indicators and the business context and enriching those indicators with his presumptions.

Step 2: Design of the indicator system

After the identification of potential indicators, the user designs the indicator system by defining relations between indicators. In some cases a relation seems to be obvious (e.g., due to 'common sense'). However, in this method the user is recommended to think about the rationale underlying this relation, consciously decide for it, and make it explicit ('presumptions'). If he concludes after all that the relation is not as obvious as it seems, he is able to expatiate his presumtions. In doing so, he fosters the maturity of the emerging indicator system (cf. refinement phase). Furthermore, the user can incorporate the related business context to identify (potential) relations between indicators in two ways:

(i) For each indicator associated to a reference object on a specific level of abstraction, the user can check for and assess potentially affected indicators by 'following' the associations of this reference object within the enterprise model. For instance, an indicator measuring a specific characteristic of a resource (e.g., 'IT hardware costs' – cf. Figure 4) might have an influence on indicators measuring the processes the resource is allocated to ('Average throughput time'). The user can then enrich the indicator system by making this relation between these two indicators explicit, i.e., associating them via a customized relationship – in this case denoting a 'bidirectional correlation' with the rationale 'subjectively estimated'. In addition, some enterprise models already express causal relationships (e.g., cause-and-effect relations of costs [He08], risks, or goals [FL07]). Here, the user can comfortably assess these relations between reference objects whether they are also applicable to indicators.

(ii) Enterprise models allow for integrating the different perspectives on a company (e.g., finance, operations, and IT). If the indicator systems of the different perspectives are all modelled in the SCORE-ML and stored in an enterprise-wide repository, indicators from other perspectives measuring the *same reference object* as well as relations to indicators in other indicator systems can be identified (e.g., further cause-and-effect relations associated with the indicator 'Turnover' (cf. Figure 4) within a financial indicator system). Thus, a multi-perspective evaluation is possible that goes clearly beyond the expressiveness of, e.g., the balanced scorecard – allowing for a tool-supported identification of potential conflicts, similarities, or need for support.

Step 3: Evaluation of the indicator system

The implementation and use of an indicator system has a substantial impact on the management of a firm. Therefore, it is mandatory to thoroughly evaluate an indicator system before using it. This is even more the case as the design of indicator systems will often involve different stakeholder with specific goals and interests. At first, the associations within an indicator system need to be analysed with respect to their consistency. The attribute 'relationSpecification' allows for analysing if an indicator might lead to actions (e.g., improving this indicator) that affect other indicators. Second, the system can be analysed for cyclic relations. Afterwards, conflicts should either be resolved or – if they are not resolvable – explicitly modelled in the indicator system. Besides this preliminary evaluation of indicator systems in the design phase, an accompanying evaluation is conducted in both the use and the refinement phase (which is omitted in Figure 5).

Use phase

The indicator systems can be used for different purposes (cf. [Gl01] p. 6; [Ge86] pp. 104ff.; [Re06] pp. 23f.), including communication between different stakeholders, reporting, and management decision support. Through its explicit and transparent representation of indicator types and relationships between them an indicator system fosters solidly founded decisions. For instance, an IT manager is guided to focus not only on his IT-related indicators but also to account for other affected business perspectives. Furthermore, an indicator systems can be used to estimate the (business-) impact of 'tuning' an indicator by analysing the effects on other indicators, goals, and even on the reference objects they measure (e.g., the various effects on a process and its other assigned indicators resulting from an optimization of its throughput time). Vice versa, the indicator systems can help to find the root-cause for a bad performing indicator. They may also help to find indicators on the operational level that serve as an instrument to achieve the desired effect on strategic level. Since underlying assumptions along with the maturity of indicators and relations are explicitly modelled ('presumption'), both can be taken into account in these analyses and, thus, should promote a better understanding and more profound decisions. As an additional feature of according tools (cf. *Req. 5*), a dashboard or other performance management software may provide navigation/retrieval functionalities that enable users to choose specific projections of instance data related to a specific indicator (e.g., filtering of instance data such as 'Average throughput time *during the last week and for product X*').

Refinement phase

During the application of the indicator systems, the users might get a more precise understanding of the indicators and their relations – especially concerning the underlying rationale and assumptions. After using the indicator system for some time, initially suspicious indicators and relations can turn out to be evident and new relations might be identified. On the other hand, some of the assumptions that influenced the design of an indicator system may turn out to be inappropriate. By adapting the indicator system and updating this information, the maturity of the indicators and, as a result, of the indicator system as a whole can increase over time.

4. Related Work

While there is a plethora of publications on business indicators, only few approaches exist that aim at modelling indicators. For instance, Pourshahid et al. [Po07] introduce a meta model for modelling KPIs on an abstract level, but they do not allow for modelling relations between them. Wetzstein et al. [We08] and Nescheuler [Ne95] suggest more elaborate approaches. However, in contrast to the claim of defining a meta model [We08] or modelling language [Ne95], they both focus on indicator instances. The main purpose of the models they target is to serve as a foundation for developing software. Hence, the resulting representations are most likely not adequate for prospective users (while the related software, on the other hand, might be; cf. *Req. 5*). [Ai97] and [Kr05] each introduce a method that accounts for the business context. Unfortunately, neither presents a language specification and – as far as one can conclude from their application examples – they only provide one type of relation between indicators (i.e., subordination). Modelling tools like ARIS or ADONIS (ADOscore) offer concepts for specifying indicators; to a certain extent, their indicator types can be assigned to concepts representing the business context. However, they usually do not support different types of relationships between indicators – as well as language specifications are not available, too.

Modelling indicators is a pivotal aspect of data warehouse modelling. In order to provide a more convenient and consistent approach to modelling multi-dimensional data warehouses, Sapia et al. [Sa99] present an extension of the EERM. Their focus is on dimensions, which express the multi-dimensional structure of data and the navigation between these dimensions (e.g., roll-up, drill-down); concepts for the business context are not provided. A number of similar approaches exist. Mansmann et al. [MNS07], for instance, present a method for transforming business process models into data warehouse models in order to add specific analytical capabilities to business process management systems (e.g., for querying multi-dimensional data). They provide a foundation for modelling data warehouses and data collections – primarily of business process instances – and can complement our method in the task of data-management. However, their approach focuses on business processes only and does not intend to support users with a more meaningful concept of indicators.

Goal modelling is a further field that is related to indicator modelling. It is a research subject both in Artificial Intelligence and in Requirements Engineering. If applied to business contexts, the basic structure of according methods is fairly similar to the one of PM methods. However, the design rationale and application objective of goal modelling methods differ decisively from PM approaches: Their main focus is on fostering requirements specification within software development processes and/or facilitation of automated reasoning based on formally specified goals (e.g., through software agents, cf. [vL01] for an overview). Common approaches include i* [Yu95], Goal Oriented Requirements Engineering (GORE, e.g., [Ju08]), and KAOS (e.g., [vL03]). Respective modelling languages offer differentiated support for formalizing goals through logical expressions. However, these languages are not well suited as a management tool, since their application requires specific expertise, and – in particular – these approaches lack support for modelling contingent subjects or circumstances that – by their nature – impede adequate formalization.

5. Evaluation, Concluding Remarks, and Future Work

In this paper we outlined a modelling method for designing and utilising business indicator systems. It aims at supporting the design of indicator systems as control instruments that conform to the goals of an enterprise and account for the relevant domain context, which is – in an ideal case – represented by an enterprise model.

The method was designed to fulfil five requirements (cf. section 2): The core concepts of the SCORE-ML have been embedded into an existing EM method that also supports modelling of processes, resources, and goals, so that it allows for associating indicators with the relevant business context (*Req. 2*). Further, we introduced the concept 'presumption' of indicators and their relations in order to expatiate the underlying rationale (*Req. 3*). Due to the integration of the SCORE-ML with other modelling languages (here: the family of MEMO languages, which explicitly encourages different perspectives and levels of abstractions in the enterprise) the method supports different perspectives on business indicator systems (e.g., from IT management and business management), and hence, allows for the integration of different indicator systems into an enterprise-wide indicator system (*Req. 1 & 4*). The semantics of the SCORE-ML allows for transforming an indicator model into, e.g., a database scheme or a class diagram in order to develop software for managing and visualizing indicators (*Req. 5*).

Compared to other approaches for describing indicators, such as 'traditional' indicator descriptions (e.g., as used for ZVEI, RL or the Balanced Scorecard) or modelling languages for designing multi-dimensional data models, our approach shows clear advantages – mainly by featuring a higher level of semantics. Table 1 shows a comparative (though preliminary) evaluation based on the identified requirements (cf. section 2) and the challenges (cf. section 3.1).

In our future research, we will focus on the advance of the method and on a tight integration with existing MEMO modelling languages. This will probably result in giving up the SCORE-ML for an integration of its concepts with other languages, such as the ORGML, the RESML or the ITML. Although the method promises to overcome the shortcomings of the current design and utilisation of indicator systems, its application is bound to certain restrictions: The task of designing an elaborate indicator system is still time-consuming and demanding – especially for untrained personnel. To foster the acceptance of the method, we plan to integrate its concepts into MEMO Center, the MEMO modelling software environment. Additionally we work on reconstructing further established indicator systems in order to build reference indicator systems – in the sense of reference models – that allow for safe and convenient reuse; the first reconstructions are available at http://openmodels.org.

Similar to other modelling methods, the proposed approach should be well suited for education purposes, e.g., for analysing and discussing existing indicator systems. Especially the explicit consideration of underlying assumptions and possible conflicts fosters a better understanding of indicator systems from business practise and the preconditions of their appropriate use.

	Traditional Indicator System	Multi Dimensional Data Model	Domain Specific Modelling Language
***Reuse:* Elaborate concept of 'indicator'**	–	o	+
***Integrity:* Protection against inconsistent models (cf. Req. 1)**	–	–	+
***Integrity:* Support for appropriate interpretations (cf. Req. 2)**	–	–	+
***Analysis:* Differentiated, formal description of indicators**	–	o	+
***Use:* Support for adequate visualization (cf. Req. 4)**	o	o	+
***Reuse:* visualization suited for indicator instances, too**	Usually no differentiation between type and instance level	–	+
***Software Development:* Concepts can be transformed into elaborate implementation level representations (cf. Req. 5)**	–	o	+

Table 1 – Comparison of approaches for indicator systems
(–: not satisfactory; o: accounted for; +: good)

Acknowledgements

We would like to thank Kirk Wilson for valuable comments on a previous version of the manuscript.

References

[Ai97] Aichele, C.: Kennzahlenbasierte Geschäftsprozessanalyse. Gabler, 1997.

[Ch76] Chen, P. P.: The Entity-Relationship Model – Toward a Unified View of Data. In: ACM Transactions on Database Systems 1 (1976) 1, pp. 9-36.

[Ec05] Eckerson, W. W.: Performance Dashboards: Measuring, Monitoring, and Managing Your Business. Wiley & Sons, Hoboken, NJ, 2005.

[EJ01] Esser, R.; Janneck, J. W.: A Framework for Defining Domain-Specific Visual Languages. In: Proceedings of the Workshop on Domain Specific Visual Languages at OOPSLA 2001, Tampa Bays, 2001.

[Fr02] Frank, U.: Multi-Perspective Enterprise Modeling (MEMO): Conceptual Framework and Modeling Languages. In: Proceedings of the Hawaii International Conference on System Sciences (HICSS-35), Honolulu, 2002.

[Fr08] Frank, U.: MEMO Meta Modelling Language (MML) and Language Architecture (revised version). ICB Research Report. Universität Duisburg-Essen. 2008. URL: http://www.icb.uni-due.de/fileadmin/ICB/research/research_reports/ICB-Report_No24.pdf

[FL07] Frank, U.; Lange, C.: E-MEMO: a method to support the development of customized electronic commerce systems. In: Inf. Syst. E-Business Management 5 (2007) 2, pp. 93-116.

[Ge86] Geiß, W.: Betriebswirtschaftliche Kennzahlen. Theoretische Grundlagen einer problemorientierten Kennzahlenanwendung. Lang, Frankfurt am Main u.a., 1986.

[Gl01] Gleich, R.: Das System des Performance Measurement : Theoretisches Grundkonzept, Entwicklungs- und Anwendungsstand. Vahlen, München, 2001.

[Gr02] Grüning, M.: Performance Measurement-Systeme: Messung und Steuerung von Unternehmensleistung. Deutscher Universitäts-Verlag, 2002.

[He08] Heise, D.; Strecker, S.; Frank, U.; Jung, J.: Erweiterung einer Unternehmensmodellierungsmethode zur Unterstützung des IT-Controllings. In (Bichler, M. et al. Eds.): Proceedings of the Multikonferenz Wirtschaftsinformatik 2008, Berlin, 2008; pp. 1017-1028.

[Ju07] Jung, J.: Entwurf einer Sprache für die Modellierung von Ressourcen im Kontext der Geschäftsprozessmodellierung. Logos, Berlin, 2007.

[Ju08] Jureta, I. J.; Faulkner, S.; Schobbens, P. Y.: Clear justification of modeling decisions for goal-oriented requirements engineering. In: Requirements Engineering 13 (2008), pp. 87-115.

[KN92] Kaplan, R.; Norton, D.: The balanced scorecard – measures that drive performance. In: Harvard Business Review 70 (1992) 1, pp. 71-79.

[Ki08] Kirchner, L.: Eine Methode zur Unterstützung des IT-Managements im Rahmen der Unternehmensmodellierung. Logos, Berlin, 2008.

[Kr05] Kronz, A.: Management von Prozesskennzahlen im Rahmen der ARIS-Methodik. In (Scheer, A.-W. et al. Eds.): Corporate Performance Management. Springer, 2005, pp. 31-44.

[Kü06] Kütz, M.: IT-Steuerung mit Kennzahlensystemen. dpunkt, Heidelberg, 2006.

[LO06] Liebetruth, T.; Otto, A.: Ein formales Modell zur Auswahl von Kennzahlen. In: Controlling 18 (2006) 1, pp. 13-23.

[Lu04] Luoma, J.; Kelly, S.; Tolvanen, J. P.: Defining Domain-Specific Modeling Languages: Collected Experiences. In: Proceedings of the OOPSLA Workshop on Domain-Specific Modeling (DSM'04), Vancouver, 2004.

[LC91] Lynch, R. L.; Cross, K. F.: Measure Up!: Yardsticks for Continuous Improvement. Wiley, Blackwell, 1991.

[MNS07] Mansmann, S.; Neumuth, T.; Scholl, M.: Multidimensional Data Modeling for Business Process Analysis. In: Proceedings of the ER 2007, Auckland, New Zealand, 2007; pp. 23-38.

[Ne95] Neuscheler, F.: Ein integrierter Ansatz zur Analyse und Bewertung von Geschaftsprozessen. Forschungszentrum Karlsruhe, Karlsruhe, 1995.

[OM07] Object Management Group: Unified Modeling Language Infrastructure. 2007. URL: http://www.omg.org/docs/formal/07-11-04.pdf (2008-07-28).

[OGC07] Office of Government Commerce (Eds.): ITIL – Service operation. The Stationery Office, London 2007.

[Po07] Pourshahid, A.; Chen, P.; Amyot, D.; Weiss, M.; Forster, A.: Business Process Monitoring and Alignment: An Approach Based on the User Requirements Notation and Business Intelligence Tools. In: Proceedings of the 10th Workshop of Requirement Engineering.(WERE'07), Toronto, 2007; pp. 80-91.

[Re06] Reichmann, T.: Controlling mit Kennzahlen und Management-Tools: Die systemgestutzte Controlling-Konzeption. Vahlen, München, 2006.

[Sa99] Sapia, C.; Blaschka, M.; Höfling, G.; Dinter, B.: Extending the E/R Model for the Multidimensional Paradigm. In: Advances in Database Technologies (LNCS 1552). Springer, 1999, pp. 105-116.

[Si90] Siegwart, H.: Kennzahlen für die Unternehmungsführung. Haupt, Bern, 1990.

[So05] Son, S.; Weitzel, T.; Laurent, F.: Designing a Process-Oriented Framework for IT Performance Management Systems. In: Electronic Journal of Information Systems Evaluation 8 (2005) 3, pp. 219-228.

[vL01] van Lamsweerde, A.: Goal-Oriented Requirements Engineering: A Guided Tour. In: 5th IEEE International Symposium on Requirements Engineering (RE 2001), 27-31 August 2001, Toronto, Canada. IEEE Computer Society, 2001, p. 249.

[vL03] van Lamsweerde, A.: From System Goals to Software Architecture. In (Bernardo, M., Inverardi, P. Eds.): Formal Methods for Software Architectures. Springer, 2003, pp. 25-43.

[Wa01] Wall, F.: Ursache-Wirkungsbeziehungen als ein zentraler Bestandteil der Balanced Scorecard: Moglichkeiten und Grenzen Ihrer Gewinnung. In: Controlling 13 (2001) 2.

[We08] Wetzstein, B.; Ma, Z.; Leymann, F.: Towards Measuring Key Performance Indicators of Semantic Business Processes. In (Abramowicz, W., Fense, D. Eds.): Proceedings of the BIS, Innsbruck, Austria, 2008; pp. 227-238.

[Yu95] Yu, E. S.-K.: Modelling strategic relationships for process reengineering. 1995.

[ZE89] Zentralverband der Elektrotechnischen Industrie: ZVEI-Kennzahlensystem. Sachon, Mindelheim, 1989.