# Reasoning with Key Performance Indicators

Daniele Barone[1], Lei Jiang[1], Daniel Amyot[2], and John Mylopoulos[1]

[1] Department of Computer Science, University of Toronto, Toronto, ON, Canada, barone/leijiang/jm@cs.toronto.edu

[2] EECS, University of Ottawa, Ottawa, ON, Canada, damyot@eecs.uottawa.ca

**Abstract.** Business organizations continuously monitor their environments, looking out for opportunities and threats that may help/hinder the fulfilment of their objectives. We are interested in strategic business models that support such governance activities. In this paper, we focus on the concept of composite indicator and show how it can be used as basic building block for strategic business models that support evaluation and decision-making. The main results of this paper include techniques and algorithms for deriving values for composite indicators, when the relationship between a composite indicator and its component indicators cannot be fully described using well-defined mathematical functions. Evaluation of our proposal includes an implemented Eclipse-based prototype tool supporting these techniques and two ongoing case studies.

**Key words:** Business intelligence, Business model, Conceptual modeling languages, Key performance indicators, Strategic planning.

## 1 Introduction

An indicator, or more precisely a Key Performance Indicator (KPI), is an industry term for a measure or metric that evaluates performance with respect to some objective. Indicators are used routinely by organizations to measure both success and quality in fulfilling strategic goals, enacting processes, or delivering products/services. For example, the indicator "Percentage increase of customer-base" may be appropriate for the goal "Increase market share", while "Average duration" might serve as indicator for the activity "Loan application".

Indicators constitute an important element of business modelling as they offer criteria for determining whether an organization is fulfilling its objectives, be they strategic goals, quality requirements, or production targets. Nowadays, they also see applications in other areas. In Requirements Engineering (RE), indicators have been used to evaluate the degree of fulfillment of goals [15], whereas in self-adaptive software systems they serve as monitored variables that determine whether a system is doing well relative to its mandate, or whether it should adapt its behaviour [14].

To choose the right indicators for a given object, be it a goal, process or product, one must have a good understanding of what is important to the organization. Moreover, this importance is generally contextual. For instance, indicators useful to a finance team may be inappropriate for a sales force. Because of the need to develop a good understanding of what is important, performance indicators are closely associated with techniques for assessing the present state of the business. A very common method for choosing indicators is to apply a management framework such as the Balanced Scorecard [8], whereby indicators measure a range of factors in a business, rather than a single one (e.g., profits).

The objects that indicators assess are generally composite, consisting of hierarchies of elements. For instance, goals are usually modelled as AND/OR tree hierarchies of sub-goals to reflect a reductionist view of problem solving. Likewise, processes are usually defined in terms of sub-processes ultimately reduced to atomic actions that an agent can perform, and products are modelled as aggregates of simpler parts that are themselves composite objects amenable to further decompositions. Alternatively – and orthogonally to the examples above – a process/product may be a root node of a taxonomy tree that defines specializations. Of course, the value of an indicator for an object should depend on the values of indicators for objects one level lower in the hierarchy. Unfortunately, there are no guidelines on what this dependency is and how to define it consistently for a given business model.

In this paper, we focus on *composite* indicators, which are indicators whose values are obtained from those of their *components*. These components themselves may also be composite, leading to a hierarchy of indicators. We are interested in the problem of propagating values of indicators from a lower level in a hierarchy to ones higher up, much like the label propagation in goal reasoning [1, 5]. This type of analysis is essential for calculating / deriving values for composite indicators. This is a non-trivial problem, since in many cases there is no well-defined mathematical function that relates component indicators to a composite one. This might simply be due to lack of knowledge about the indicators, or the intrinsic nature of the indicators at hand. The main contributions of this paper include: i) different techniques and algorithms for deriving values of composite indicators, especially when the relationship between a composite indicator and its components cannot be fully described using well-defined mathematical functions, and ii) an Eclipse-based prototype tool supporting these techniques. In particular, this is an extended and improved version of [2], with additional material and examples on how to reason with composite indicators.

This research is conducted in the context of the Business Intelligence Network, a Canada-wide strategic research network. Our long-term objective within the network is to develop a conceptual modelling language, called Business Intelligence Model (BIM) [3], for modelling business objectives, processes and objects in order to support business intelligence activities.

The rest of the paper is structured as follows. Sections 2 presents key concepts for strategic business models. Section 3 introduces the concept of indicator and how it can be used to evaluate goals and situations. Section 4 presents three techniques to derive values of composite indicators using different estimation / approximation techniques. Section 5 briefly presents an Eclipse-based prototype tool that supports these techniques. Finally, Sections 6 and 7 discuss related work and conclusions, respectively.

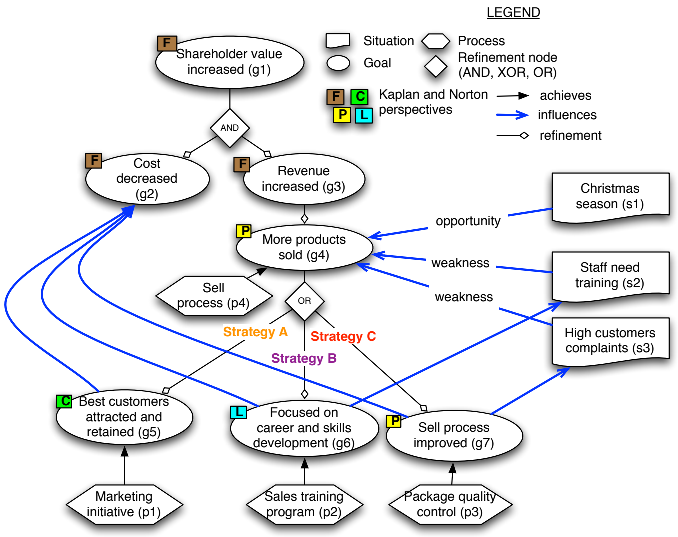

LEGEND
Situation
Goal
Process
Refinement node (AND, XOR, OR)
Kaplan and Norton perspectives
achieves
influences
refinement
Shareholder value increased (g1)
AND
Cost decreased (g2)
Revenue increased (g3)
More products sold (g4)
Christmas season (s1)
Staff need training (s2)
High customers complaints (s3)
opportunity
weakness
weakness
Sell process (p4)
OR
Strategy A
Strategy B
Strategy C
Best customers attracted and retained (g5)
Focused on career and skills development (g6)
Sell process improved (g7)
Marketing initiative (p1)
Sales training program (p2)
Package quality control (p3)

**Fig. 1.** Examples of goals, situations and influences.

## 2 Strategic Business Models

In this section, we review some of the key concepts used in BIM [3] to support strategic business modelling and reasoning about strengths, weaknesses, opportunities and threats (popularly known as SWOT). Technical details about these concepts, including semantics, are presented in [7].

A *goal* (also intention, objective, vision, mission) represents a desired state-of-affairs, defined during strategic planning and pursued (hopefully successfully) during business operation. The most basic characteristics of goals include: i) a goal may be AND/OR-refined into subgoals so that its *satisfaction* level depends on that of its subgoals; ii) a goal may be satisfied in more than one way if it or any of its refinements are OR-refined, in which case a choice needs to be made among alternative subgoals in deciding how to fulfill the root-level goal; and, iii) a goal's satisfaction may be affected by that of goals other than its subgoals. Goal analysis produces a goal model consisting of an AND/OR refinement tree with additional positive/negative *contributions*. The satisfaction level of a goal can be inferred from that of others in the same goal model using a label propagation algorithm [5, 1]. Examples of goals are shown in Figure 1. Notice how the "Shareholder value increased" goal is AND-decomposed into the sub-goals "Cost decreased" and "Revenue increased"; similarly, the "More Products sold" goal is OR-decomposed in three different alternative sub-goals (strategies), namely "Best customers attracted and retained", "Focused on career and skills development", and "Sell process improved". An example of influence among goals is represented by the one existing from the "Sell process improved" goal towards the "Cost decreased" goal.

In addition to goals, we model partial states of the world as *situations*. For strategic business models, we need the notion of organizational situation, such as "Christmas season", an opportunity for a sales organization, or "Competitor buys key technology", a potential threat. Analogously to satisfaction levels for goals, we have *occurrence* levels for situations, which define the degree to which a situation occurs in the current state-of-affairs. The situations "Christmas season", "Staff need training", and "High customers complaints" described in Figure 1, are some examples of partial states of the world that can occur within a business context.

To reason about goal satisfaction under the influence of situations, we extend the contribution relation so that it can be used to relate any combination of goals and situations. Hereafter, we refer to it with the term of *influence*. For example, the situation "Christmas season" positively influences the goal "Increase sales", while the situation "Booming economy" positively influences the situation "Growing inflation". Figure 1 shows some examples of such influences, e.g., the "Staff need training" situation, representing an internal weakness for the company, influences negatively the "More products sold" goal.

We characterize influences along two dimensions: i) *direction*: a positive (resp. negative) influence exists from a situation/goal to another if the occurrence/satisfaction of the source increases (resp. decreases) the occurrence/satisfaction of the target; and, ii) degree or *strength*: an influence is full if it is a causal relation (i.e., 100% effect on the target of the goal or situation influenced); otherwise, it is partial.

## 3 Indicators

An *indicator* is a measure, quantitative or qualitative, of the progress or degree of fulfillment of organization goals. The subject of an indicator is a particular *feature* or *quality* of an element in the business environment, e.g., the *workload* of an employee, or the *compliance* of an internal process with respect to external regulations.

To express *why* an indicator is needed and *what* it is measuring, we rely on two relations, *evaluates* and *measures*, as illustrated in Figure 2. In the example, the indicator "Number of products sold" is needed (why) to evaluate the goal "More products sold" by measuring the task "Sell products".

Each indicator, being a composite or component, has a *current value* which is is evaluated against a set of parameters: *target (value)*, *threshold (value)* and *worst (value)* [10]. The result of such evaluation is a normalized value (ranging in $[-1, 1] \subset \mathbb{R}$), which is often referred to as the *performance level*. Note that a current value can be assigned by either: i) extracting it at run-time from back-end data sources[1], ii) supplied by users to explore "what-if" scenarios, or iii) calculated by a *metric* expression (see Section 4.1).

[1] *Dimensions* and *levels* [10] can be used to filter data from datawarehouses.

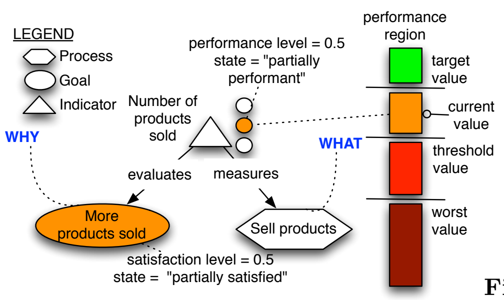


Fig. 2. An example of an indicator to evaluate a goal.

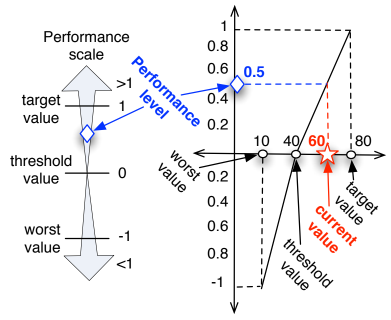


Fig. 3. Example of interpolation [12] to calculate performance levels.

An indicator can be *positive*, *negative*, or *bidirectional*, meaning that we want to maximize, minimize or balance its target. *Performance regions* are defined for each type of indicator by properly combing the indicator's parameters. Figure 2 shows an example of performance region for a positive indicator, i.e., we want to maximize the number of products sold, in which $Target \geq Threshold \geq Worst\ value$.

The relative position of indicator current values within such regions leads to indicator performance levels, as shown by Figure 3. Notice how the worst and target values are mapped respectively to -1 and +1 levels, while the threshold value is mapped to 0. A linear interpolation[2] is used to approximate performance levels, as also described by System equation 1 [12]. For instance, the performance level (pl) for Figure 3 is $pl(60) = \frac{|60-40|}{|80-40|} = 0.5$.

$$pl(current\ v.) = \begin{cases} \frac{|current\ v. - threshold\ v.|}{|target\ v. - threshold\ v.|}, & \text{if } current\ v. \geq threshold\ v. \\ -\frac{|current\ v. - threshold\ v.|}{|threshold\ v. - worst\ v.|}, & \text{if } current\ v. < threshold\ v. \end{cases} \quad (1)$$

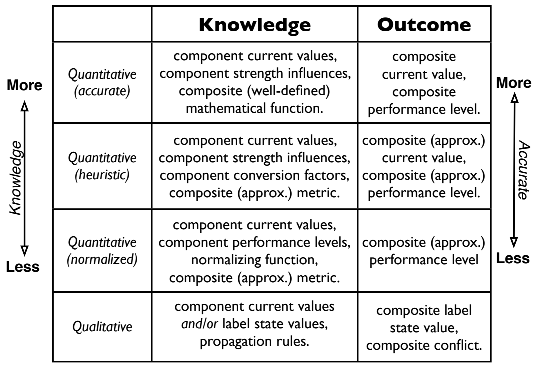
Performance levels are, in turn, propagated to the corresponding goals to evaluate satisfaction levels. For example, in Figure 2, the performance level 0.5 is propagated to the satisfaction level of the corresponding goal which, in turn, is mapped to a "partial satisfied" state (orange colour[3]).

As we will show in the following section, indicators can be used to evaluate situations in a similar way we do for goals, by propagating a performance level to the occurrence level of the situation under evaluation. For example, the indicator "Number of products returned" can evaluate the situation "Low number of returned products".

[2] Other forms of interpolation can be used, e.g., polynomial, spline, etc.
[3] BIM provides mapping tables to map satisfaction, occurrence and performance levels to corresponding states of business elements.

## 4 Reasoning with Indicators

In a business model, indicators are associated with various business elements in the model. These elements in general are composite, consisting of hierarchies of elements. Such structure implies hierarchies for indicators. For example in Figure 6, the goal hierarchy results in a hierarchy for associated indicators. More specifically, "Number of special package" is a component indicator of "Number of products sold", since it evaluates the goal "Sell process improved" which is a sub-goal of "More products sold".



| | **Knowledge** | **Outcome** |
|---|---|---|
| *Quantitative (accurate)* | component current values, component strength influences, composite (well-defined) mathematical function. | composite current value, composite performance level. |
| *Quantitative (heuristic)* | component current values, component strength influences, component conversion factors, composite (approx.) metric. | composite (approx.) current value, composite (approx.) performance level. |
| *Quantitative (normalized)* | component current values, component performance levels, normalizing function, composite (approx.) metric. | composite (approx.) performance level |
| *Qualitative* | component current values *and/or* label state values, propagation rules. | composite label state value, composite conflict. |

**Fig. 4.** Techniques classification.

We are interested in algorithms that propagate values of indicators from a lower level in a hierarchy to ones higher up. We classify such propagation into four categories, as described in Figure 4, based on what is known about the relation between a composite indicator and its components. In the simplest case, such a relation is fully described using a mathematical function, and there is no problem in computing values for the composite indicator. For example, profits can be calculated directly from revenues and costs. In other cases, when such a mathematical relation does not exist, indicator values have to be derived using estimation/approximation techniques.

In what follows, we present three techniques to derive values of composite indicators using conversion factors, range normalization, and qualitative reasoning. An end-user may prefer one or the others depending on the quantity of information of the domain she/he posses, or on the available time she/he has for encoding such an information into the model. The qualitative approach may also be chosen when the user is interested in an early analysis of conflicts and inconsistencies within the same goals of a strategy.

### 4.1 Deriving Composite Indicators Using Conversion Factors

When a composite indicator does not share that same unit of measure with its components, a necessary condition for finding a metric that computes its values is that there is a suitable conversion factor for each component indicator that has a different unit of measure.

For example, consider the two indicators "Employee cost" and "Working time". In particular, Employee cost can be defined as a composite indicator whose value relies on the component indicator Working time. According to our requirement, we need to convert the current value of Working time values measured in hours into Employee cost units. One possible conversion factor is to take the average of the wage per hour for all employees[4]. Assuming that such an average is 20 and that the current value for Working time is 160 hours, we can calculate an approximated current value for Employee cost as:

1. $20 \text{ dollars} = 1 \text{ hours} \rightarrow 20 \frac{\text{dollars}}{\text{hours}} = 1$, where 20 is the conversion factor;
2. $160 \text{ hours} \cdot 20 \frac{\text{dollars}}{\text{hours}} = 3,200 \text{ dollars}$.

Notice that in many cases a conversion factor is an estimate based on previous experience / statistics. For example, the average wage per hour could be 30 instead of 20 for a different company. When conversions are impossible, e.g., it is not possible to convert gallons to square feet, we have to fall back to a "normalized" approach or to a "qualitative" one; these are presented, respectively, in Sections 4.2 and 4.3.

When conversion factors are available, we become able to define valid metric expressions that contain: i) current values for component indicators, ii) influence strengths, and iii) conversion factors. With this aim, we adopt and use off-the-shelf the grammar of the Jep Java Library (see Section 5), which allows us to express rich and flexible expressions to meet user requirements.

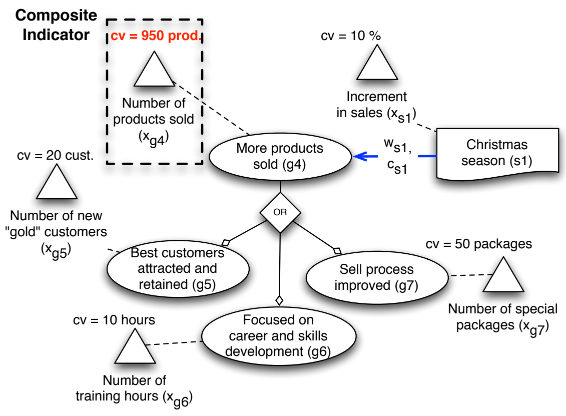

Composite Indicator
cv = 950 prod.
Number of products sold ($x_{g4}$)
cv = 10 %
Increment in sales ($x_{s1}$)
$w_{s1}$, $c_{s1}$
Christmas season (s1)
More products sold (g4)
cv = 20 cust.
Number of new "gold" customers ($x_{g5}$)
OR
Best customers attracted and retained (g5)
Sell process improved (g7)
cv = 50 packages
Number of special packages ($x_{g7}$)
cv = 10 hours
Focused on career and skills development (g6)
Number of training hours ($x_{g6}$)

**Fig. 5.** Examples of reasoning with conversion factors: cv = current value, w = weight, c = conversion factor.

An example of such an expression is $x_{g_4} = x^e_{g_4} + w_{s_1} \cdot c_{s_1} \cdot x_{s_1} + \sum_{j=5}^{j=7} w_{g_j} \cdot c_{g_j} \cdot x_{g_j}$, which is used in Figure 5 to calculate the current value of the "Number of products sold" indicator. In the expression, $x^e_{g_4}$ is the expected value of products sold while the different $w_m$ and $c_n$ are, respectively, influence strengths and conversion factors of the component indicators. When the designer defines an expression, she/he must consider two kinds of factors (where by "factors" we mean some quantity which can influence positively or negatively the final value of a composite indicator). First, she/he must take into account those factors derived

[4] This value can be also defined as an "Average hourly wage" indicator and, in turn, as a related indicator for the Employee cost.

from "influencers". In the previous example, we have the situation "Christmas season", which influences and contributes positively to the composite indicator with the quantity $w_{s_1} \cdot c_{s_1} \cdot x_{s_1}$. In particular, $x_{s_1}$ is the current value (10 %) of the indicator "Increment in sales", $w_{s_1}$ and $c_{s_1}$ are, respectively, the strength of the influencer, and the conversion factor used to convert the 10 % increment value into a number of products sold. These last two parameters must be chosen accurately by the designer who must rely on her/his domain experience and/or estimates of historical data.

The second type of factors considered are those derived from sub-goals. For example, in Figure 5, the component indicators associated with the sub-goals "Best customers attracted and retained", "Focused on career and skills development" and "Sell process improved" all contribute to the composite indicator expression with a total quantity of $\sum_{j=5}^{j=7} w_{g_j} \cdot c_{g_j} \cdot x_{g_j}$. In this case, a sum operator was chosen to aggregate such quantity (i.e., products sold) derived from the achievement of one or more of these sub-goals. In fact, the achievement of the "Best customers attracted and retained" sub-goal increments the number of new "gold" customers, which in turn represents an increment of products sold. A gold customer buys an average of 5 products during intense sales seasons, and such information can be used by the designer to tune the conversion factor used in the above expression.

The two sets of factors, one from influencers, the other from sub-goals, are then aggregated together to compute a final value for the composite indicator. Again, a sum operator was used, but the designer can customize each expression depending on the semantic of influencers and sub-goals, and how their values may impact the final value of the composite indicator.

## 4.2 Deriving Composite Indicators Using Range Normalization

When conversion factors are not available, a way to derive composite indicators is to rely on range normalization, which takes values spanning a specific range and represents them in another range. Range normalization is applicable when we do not need to obtain the exact value for a composite indicator, but rather only its performance level. Indeed, when we calculate the performance level of an indicator (by using its current value and parameters as described in Section 3), we are producing a "normalized value" in a range within $[-1, +1] \subset \mathbb{R}$.

A performance level for a composite indicator is calculated by combining performance levels of component indicators. We propose a BNF grammar that allows to define rules for combining levels of component indicators into the one of a composite one. The grammar, presented in Table 1, is highly expressive and can be extended to accommodate new combination rules. Notice how the grammar builds a metric expression by combing (row 3 : $combInfDec$) performance levels propagated from influencers (row 4 : $plInf$) and from sub-nodes (row 5 : $plDec$). The grammar also embeds the concept of "tolerance" [1], which allows to limit the *total* (negative or positive) influence of influencers to the node at hand.

This approach is applied in Figure 6 in which no conversion factors are available. By relying on the grammar, a metric expression was defined for

**(E)BNF Grammar for Performance Level Metric Expressions**

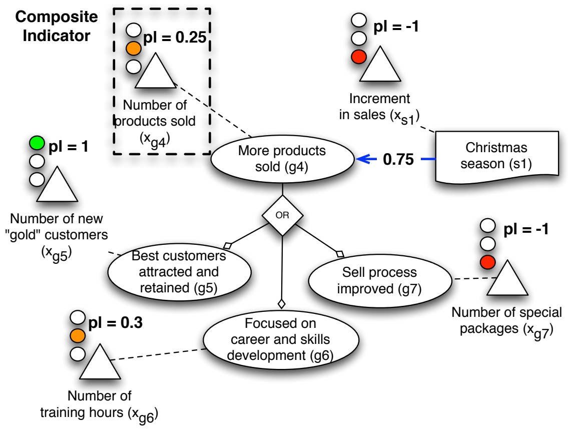
```
         Start = plExp;

         plExp = 'pl(' , Identifier , ') = ' , (combInfDec | plInf | plDec) , ')';

    combInfDec = Function , '(' , plInf , ',' , plDec , ')' , {Operator , (Identifier | FN)};
         plInf = [Function] , InfluencersList , ['{'Tolerance'}'];
         plDec = [Function] , Sub-nodesList , ['{'Tolerance'}'];

 Sub-nodesList = '[' , Sub-node , {',' Sub-node} , ']';
InfluencersList = '[' , Influencer , {',' Influencer} , ']';
     Tolerance = '1' | '0' | ('0.' , DL);
      Sub-node = 'pl(' , Identifier , ')';
    Influencer = Weight ,'·' , 'pl(' , Identifier , ')';
        Weight = Identifier;
      Function = ('sum' | 'sum^t' | 'min' | 'max' | 'avg');
      Operator = ('+'| '-'| '*'| '/');
    Identifier = AC , {AC | D};
            AC = 'A' | 'a' | 'B' | 'b' | ... | 'Z' | 'z';
            FN = DL | ( DL , '.' , DL);
            DL = D | (D , DL);
             D = '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9';
```

**Table 1.** A BNF grammar for performance level metric expressions. We use the EBFN ISO notation for symbols: definition(=), concatenation (,), termination(;), alternative(|), optional([...]), repetition({...}), grouping((...)), terminal string ('...').



Composite Indicator
pl = 0.25
Number of products sold ($x_{g4}$)
pl = -1
Increment in sales ($x_{s1}$)
pl = 1
Number of new "gold" customers ($x_{g5}$)
More products sold (g4)
0.75
Christmas season (s1)
OR
Best customers attracted and retained (g5)
Sell process improved (g7)
pl = -1
Number of special packages ($x_{g7}$)
pl = 0.3
Focused on career and skills development (g6)
Number of training hours ($x_{g6}$)

**Fig. 6.** Examples of reasoning with performance levels of indicators.

the composite indicator "Number of products sold": $pl(x_{g4}) = sum^t\{0.75 \cdot pl(x_{s1}), max[pl(x_{g5}), pl(x_{g6}), pl(x_{g7})]\}$. The metric takes the maximum performance level among the sub-nodes, and sums such a level to the result obtained by multiplying the influence strength 0.75 by the performance level of the component indicator "Increment in sales". The special operator $sum^t$ allows one to normalize the final result in the range $[-1, +1] \subset \mathbb{R}$, to be further used in other computations. In the example, the result is: $sum^t\{0.75 \cdot -1, max[1, 0.3, -1)]\} = 0.25$ (if such a value was greater than 1 or lower than -1, it would have been normalized to 1 or -1, respectively).

We have defined a BNF grammar for determining the exact syntax for our expression's language and find all available options for any expression. Notice that, since the BNF here presented could encompass the expression of computations described in Section 4.1, we are considering to extend the Jep Java Library to satisfy such a goal.

A BNF grammar consists of a set of "non-terminals" and "terminals". Non-terminals are placeholders within a BNF definition, defined elsewhere in the BNF grammar. For example, the non-terminal "Function" that appears in the third row of Table 1 is defined some rows later by the set of terminals: { **'sum'** | **'sum$^t$'** | **'min'** | **'max'** | **'avg'** }. Terminals are endpoints in a BNF definition consisting, in our case, of keywords representing functions (as above) and operators (**+**, **-**, *****, **/**), lower (**a**–**z**) and upper case (**A**–**Z**) alphabet characters, and numbers (**0**–**9**) (in the grammar, all terminals appear in bold type).

Designers can define an expression starting from the non-terminal *Start* and recursively substituting each non-terminal (present on the right side of Table 1), with the appropriate definition, until all the non-terminals are (re)defined by terminals. For example, the non-terminal *Start* is defined by the non-terminal *plExp* which is (re)defined by the expression **'pl('** Identifier **') = '** (combInfDec | plInf | plDec) **')'**. The symbol "|" allows for alternatives, therefore the designer must take here her/his first design decision.

Suppose that she/he wants to define the performance level of an indicator *A1*, which is decomposed in two sub-indicators *A2* and *A3* (i.e., we have a graph with a goal G1 with two sub-goals G2 and G3).

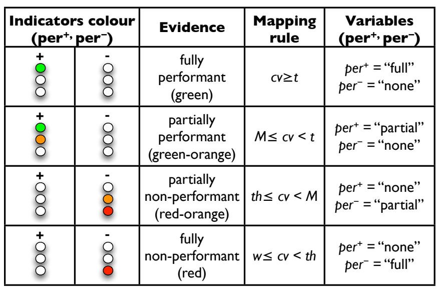
She/he must choose the alternative *plDec*, which states for "decomposition", and:

1. (re)define the non-terminal *Identifier* in the non-terminals *AC(D)* and, in turn, in the terminals **A** and **1** to represent the indicator *A1*;
2. (re)define the non-terminal *plDec* in the non-terminals *Function Sub-nodesList* which, with further steps, result into a terminal expression **min[pl(A2), pl(A3)]**.
3. combine all together to obtain the final expression **pl(A1) = min[pl(A2), pl(A3)]** which states that the performance level of *A1* is the minimum among the levels of *A2* and *A3*. In goal reasoning [5], this semantic is often associated with an AND-decomposition where the satisfaction of the father goal is equal to the minimum satisfaction value of its sons.

## 4.3 Deriving Composite Indicators Using Qualitative Reasoning

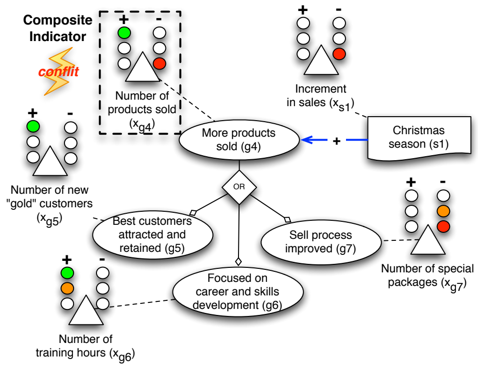
Inspired by [5], we augment the techniques of sections 4.1 and 4.2, with a qualitative reasoning technique. In this case, instead of propagating indicator performance levels, we propagate the categorical label assigned to them. This technique has long been used for qualitative goal reasoning in RE.

A major difference between this technique and the ones presented in previous sections, is that conflicts are allowed, i.e., an indicator can be at the same



| Indicators colour (per⁺, per⁻) | | Evidence | Mapping rule | Variables (per⁺, per⁻) |
|---|---|---|---|---|
| + 🟢 ⚪ ⚪ | - ⚪ ⚪ ⚪ | fully performant (green) | $cv \geq t$ | $per^+$ = "full" $per^-$ = "none" |
| + 🟢 🟠 ⚪ | - ⚪ ⚪ ⚪ | partially performant (green-orange) | $M \leq cv < t$ | $per^+$ = "partial" $per^-$ = "none" |
| + ⚪ ⚪ ⚪ | - ⚪ 🟠 🔴 | partially non-performant (red-orange) | $th \leq cv < M$ | $per^+$ = "none" $per^-$ = "partial" |
| + ⚪ ⚪ ⚪ | - ⚪ ⚪ 🔴 | fully non-performant (red) | $w \leq cv < th$ | $per^+$ = "none" $per^-$ = "full" |

**Fig. 7.** Mapping rules, where: cv = current value, th = threshold value, w = worst value, and M = middle value among the target $t$ and threshold $th$.



Composite Indicator
conflit
+
-
Number of products sold ($x_{g4}$)
+
-
Increment in sales ($x_{s1}$)
Christmas season (s1)
+
More products sold (g4)
+
-
Number of new "gold" customers ($x_{g5}$)
OR
Best customers attracted and retained (g5)
Sell process improved (g7)
+
-
Number of special packages ($x_{g7}$)
+
-
Focused on career and skills development (g6)
Number of training hours ($x_{g6}$)

**Fig. 8.** Examples of qualitative reasoning.

time "fully performant" and "fully non-performant" (see Figure 7). This is analogously to [5], where goals have *satisfiability* values but also *deniability* ones: during label propagation, a goal can be both "partially satisfied" and "partially denied".

| | $(I_2^r, I_3^r) \overset{and}{\longmapsto} I_1^c$ | $I_2^r \overset{+S}{\longmapsto} I_1^c$ | $I_2^r \overset{-S}{\longmapsto} I_1^c$ | $I_2^r \overset{++S}{\longmapsto} I_1^c$ | $I_2^r \overset{--S}{\longmapsto} I_1^c$ |
|---|---|---|---|---|---|
| $per^+(I_1^c)$ | $min \begin{cases} per^+(I_3^r) \\ per^+(I_2^r) \end{cases}$ | $min \begin{cases} per^+(I_2^r) \\ P \end{cases}$ | $N$ | $per^+(I_2^r)$ | N |
| $per^-(I_1^c)$ | $max \begin{cases} per^-(I_3^r) \\ per^-(I_2^r) \end{cases}$ | $N$ | $min \begin{cases} per^+(I_2^r) \\ P \end{cases}$ | N | $per^+(I_2^r)$ |

**Table 2.** Propagation rules in the qualitative framework. The (`or`), (+D), (-D), (++D), (- -D) cases are dual w.r.t. (`and`), (+S), (-S), (++S), (- -S) respectively. See [5], for details.

We associate to each indicator $I_i$ two variables: positive performance ($per^+$) and negative performance ($per^-$). Ranging in {F,P,N} ("full", "partial", "none"), such that $F > P > N$, these variables represent the current evidence of performance or non-performance of an indicator $I_i$. For instance, $per^+(I_i) \geq P$ states that there is at least partial evidence that $I_i$ is performant. To assign "evidence" and, therefore, values to the variables $per^+$ and $per^-$, we use the mapping rules described in Figure 7. For example, when the current value of an indicator $I_i$ lies among its target value and the middle point M (a value which is equal distant from the target $t$ and from the threshold $th$), we have that $per^+ =$ "*partial*″ and $per^- =$ "*none*″.

Propagation of the values from component indicators to a composite indicator relies on the axioms and (adapted) propagation rules from [5], which are summarized in Table 2. For example, in Table 2, the rule $(I_2^r, I_3^r) \overset{and}{\longmapsto} I_1^c$ states how labels are propagated when there is an AND-decomposition relation between goal $G_1$ and sub-goals $G_2$ and $G_3$ (here we refer to goal nodes, but they can also be situation nodes, or a mix of both), with associated indicators $I_1^c$, $I_2^r$ and $I_3^r$. Analogously, $I_2^r \xmapsto{-S} I_1^c$ states how labels are propagated when there exists an influence relation between goals $G_2$ and $G_1$, with associated indicators $I_2^r$ and $I_1^c$. The strength of this influence is equal to $-S$, which means that if $G_2$ is satisfied, then there is some evidence that $G_1$ is denied, but if $G_2$ is denied, then nothing is said about the satisfaction of $G_1$, see [5] for further details.

Figure 8 provides an example of our qualitative approach. For each indicator, the $per^+$ variable is represented by a traffic light with a plus symbol on the top, a minus symbol is used for the $per^-$ variable. The colours for the traffic lights are those described in Figure 7. A conflict is discovered for the composite indicator "Number of products sold" when values are propagated following the rules in Table 2. When such conflicts appear in a schema, although undesirable, they do help to highlight particular aspects of a business that need user attention because of possible inconsistencies. In the following, we explain in detail how such conflict is calculated.

For each indicator, the corresponding current value is extrapolated from the data sources. For example, for the indicator "Number of training hours" we have a current value of 16. The associated target and the threshold values are, respectively, 20 and 10; therefore, by following the second mapping rule of Figure 7 ($M \leq cv < t$), we obtain that the indicator "Number of training hours" is "partially performant (green-orange)" (as also reported in Figure 8).

By applying the same procedure for all the indicators (with the exception of the "Number of products sold"), we obtain: a fully performant "Number of new "gold" customers (green)"; a partially non-performant "Number of special packages (red-orange)"; and, a non-performant "Increment in sales". The next step is to rely on the propagation rules described in Table 2 to propagate and calculate the colour of the "Number of products sold" indicator.

We start to propagate the three indicators associated to the corresponding sub-goals by relying on the rules in the first column of Table 2. Indeed, the OR rules are dual with respect to the AND rules [5]. Therefore, for the $per^+$ and $per^-$ variables, we need to choose respectively the *maximum* and the *minimum* among the values of the sub-indicators.

The result is:

$$\begin{cases} per^+(\text{from-sub-nodes}) = max[per^+(x_{g5}), per^+(x_{g6}), per^+(x_{g7})] = \text{“Full” – i.e., fully performant} \\ per^-(\text{from-sub-nodes}) = min[per^-(x_{g5}), per^-(x_{g6}), per^-(x_{g7})] = \text{“None”} \end{cases}$$

We use "from-sub-nodes" as a temporary node to store the result; in fact, we need to combine such values with the ones from the influencer "Christmas season". To propagate the associated indicator "Increment in sales" we must use the rules in second column of Table 2. Notice that, the influence from the "Christmas season" situation to the "More products sold" has a plus (+) symbol. As described in [5], this is a symmetric relation and it is a shorthand for the combination of the two corresponding asymmetric relationships $I_2^r \xmapsto{+S} I_1^c$ and $I_2^r \xmapsto{+D} I_1^c$ (the propagation rule for the latter is dual w.r.t. the former); this means that both satisfiability and deniability are propagated. Therefore, after propagation, we obtain:

$$per^{+}(\text{from-influencer}) = min \begin{cases} per^{+}(x_{s1}) = \text{“None”} \\ P \end{cases} = \text{“None”}$$

$$per^{-}(\text{from-influencer}) = max \begin{cases} per^{-}(x_{s1}) = \text{“Full”} \\ P \end{cases} = \text{“Full” – i.e., fully non-performant}$$

Again, we use a temporary node, namely “from-influencer”, to temporary store the result which must be combined with the result in the node “from-sub-nodes”. Giorgini et al. [5] provide an algorithm to combine such results, from sub-nodes (our node “from-sub-nodes”) and from influencers (our node “from-influencer”). In our case, the result is the one depicted in Figure 8 where a conflict is discovered. The main idea behind such an algorithm, is to take, for the $per^{+}$ variable of the composite indicator (i.e., the “Number of products sold” ), the maximum value among all the temporary $per^{+}$ values (both sub-nodes and influencers). The same must be done for the $per^{-}$ variable.

## 5 BIM’s Tool Support

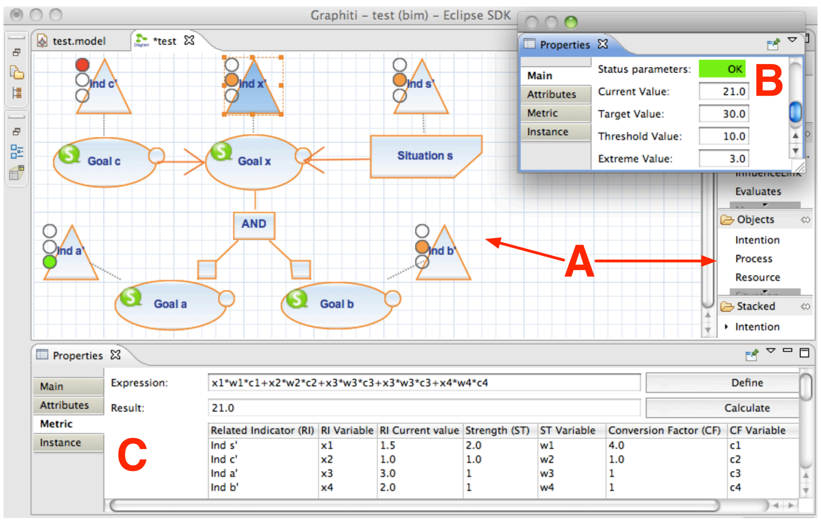
We have implemented a visual editor prototype to draw business schemas and reason about them. Our implementation uses *Graphiti*[5], an Eclipse-based graphics framework that enables easy development of state-of-the-art diagram editors for domain-specific modeling languages. The current version of the prototype implements the quantitative approach described in Section 4.1 by relying on Jep[6], a Java library for parsing and evaluating mathematical expressions. Jep supports strings, vectors, complex numbers and boolean expressions.

Figure 9 provides a snapshot of the tool. Marker “A” highlights the business schema and the toolbar containing business element constructs. Marker “B” highlights the property panel containing indicator parameters and current value. Marker “C” highlights the property panel containing the definition of the metric expression (notice available variables such as strengths, conversion factors, etc.).

## 6 Related work

The use of business-level concepts—such as goals, processes and resources—has been researched widely for at least 15 years and is already practiced to some extent in both Data Engineering and Software Engineering. In the literature, different modeling proposals exist that are related to our work, such as i* [16], URN/GRL [6] and KAOS [4, 15], all from the general area of Goal-Oriented Requirements Engineering. From these we have adopted intentional and social concepts. However, these models lack primitive constructs for influence relationships, (composite) indicators, and various types of situations integrated in the BIM modeling framework. Recent proposals have extended URN to include indicators [13]. We share ideas with this work; however: i) our indicators are more

[5] http://www.eclipse.org/graphiti/
[6] http://www.singularsys.com/jep/



Status parameters: OK
Current Value: 21.0
Target Value: 30.0
Threshold Value: 10.0
Extreme Value: 3.0

Expression: x1*w1*c1+x2*w2*c2+x3*w3*c3+x3*w3*c3+x4*w4*c4
Result: 21.0

| Related Indicator (RI) | RI Variable | RI Current value | Strength (ST) | ST Variable | Conversion Factor (CF) | CF Variable |
|---|---|---|---|---|---|---|
| Ind s' | x1 | 1.5 | 2.0 | w1 | 4.0 | c1 |
| Ind c' | x2 | 1.0 | 1.0 | w2 | 1.0 | c2 |
| Ind a' | x3 | 3.0 | 1 | w3 | 1 | c3 |
| Ind b' | x4 | 2.0 | 1 | w4 | 1 | c4 |

**Fig. 9.** Graphiti visual editor.

general and they can be used to measure any model object, including other indicators; ii) we provide more guidelines to distinguish "what" is measured and "why" it is measured; and iii) our indicators can be used to evaluate situations which, from our perspective, are fundamental for strategic reasoning.

In [11], the authors propose a formal framework for modelling goals (and for evaluating their satisfaction) based on performance indicators. Our work shares similar intentions but focuses more on the concept of composite indicator and a way to define metric expressions to calculate their values.

From a business perspective, our business schemas can capture what is commonly found in Strategic Maps [9] and Balanced Scorecards [8], but we also support reasoning and we include the concept of situation, a fundamental concept for supporting SWOT analysis. In fact, as we show in [7], we can map our approach to SWOT analysis and other languages that enable goal analysis techniques [5, 6], including probabilistic ones.

## 7 Conclusions

In this paper, we presented a model-based approach to design and reason about an organization's business environment and strategies, with a focus on Key Performance Indicators and indicator composition in the context of the Business Intelligence Model language. We provided qualitative and quantitative techniques to analyze the impact of strategies on organization goals, by relying on different types of knowledge measured through indicators. An Eclipse-based prototype was used to support our findings and validate the feasibility of our approach.

We argue that the indicators and composition mechanisms proposed here are more flexible and general than what is commonly found in related work.

As future work we are carrying out a real-world case study to empirically evaluate our composite indicator approach. We are currently involved in a Business Intelligence project at a Toronto-area hospital, where we are building a global picture of patient flow in order to identify sources of bottlenecks within and beyond the hospital (e.g., long-term care facilities, home care services, etc.).

## References

1. D. Amyot, S. Ghanavati, J. Horkoff, G. Mussbacher, L. Peyton, and E. S. K. Yu. Evaluating goal models within the goal-oriented requirement language. *Int. J. Intell. Syst.*, 25(8):841–877, 2010.
2. D. Barone, L. Jiang, D. Amyot, and J. Mylopoulos. Composite indicators for business intelligence. *ER 2011, 30th Int. Conf. on Conceptual Modeling (short paper, to appear)*, 2011.
3. D. Barone, J. Mylopoulos, L. Jiang, and D. Amyot. Business Intelligence Model, ver. 1.0. Technical Report CSRG-607 (ftp://ftp.cs.toronto.edu/csri-technical-reports/INDEX.html), University of Toronto, March 2010.
4. A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Sci. Comput. Program.*, 20(1-2):3–50, 1993.
5. P. Giorgini, J. Mylopoulos, E. Nicchiarelli, and R. Sebastiani. Reasoning with goal models. In *Proc. of the 21st Intern. Conf. on Conceptual Modeling*, pages 167–181, 2002.
6. Intern. Telecom. Union: Recommendation Z.151 (11/08). User Requirements Notation (URN) – Language definition. http://www.itu.int/rec/T-REC-Z.151/en.
7. L. Jiang, D. Barone, D. Amyot, and J. Mylopoulos. Strategic models for business intelligence: Reasoning about opportunities and threats. *ER 2011, 30th Int. Conf. on Conceptual Modeling (short paper, to appear)*, 2011.
8. R. S. Kaplan and D. P. Norton. *Balanced Scorecard: Translating Strategy into Action.* Harvard Business School Press, 1996.
9. R. S. Kaplan and D. P. Norton. *Strategy maps: Converting intangible assets into tangible outcomes.* Harvard Business School Press, 2004.
10. D. Parmenter. *Key Performance Indicators.* John Wiley & Sons, 2007.
11. V. Popova and A. Sharpanskykh. Formal modelling of organisational goals based on performance indicators. *Data & Knowledge Engineering*, 70(4):335–364, 2011.
12. A. Pourshahid, D. Amyot, L. Peyton, S. Ghanavati, P. Chen, M. Weiss, and A. J. Forster. Business Process Management with the User Requirements Notation. *Electronic Commerce Research*, 9(4):269–316, 2009.
13. A. Pourshahid, G. Richards, and D. Amyot. Toward a goal-oriented, business intelligence decision-making framework. *5th Int. MCETECH Conf. on eTechnologies, Les Diablerets, Switzerland*, 78:100–115, 2011.
14. V. Souza, A. Lapouchnian, and J. Mylopoulos. System identification for adaptive software systems: a requirements-engineering perspective. *ER 2011, 30th Int. Conf. on Conceptual Modeling (to appear)*, 2011.
15. A. van Lamsweerde. Conceptual modeling: Foundations and applications. chapter Reasoning About Alternative Requirements Options. Springer-Verlag, 2009.
16. E. Yu. Towards modelling and reasoning support for early-phase requirements engineering. In *RE'97*, pages 226–235, Washington, USA, 1997.