KPIs and Their Properties Defined with the EXTREME Method

Ella Roubtsova[1] and Vaughan Michell[2]

[1] Open University of the Netherlands. 6401DL, Valkenburgerweg 177
Heerlen, The Netherlands
[2] Henley Business School, University of Reading,
Whiteknight, Reading, RG6 6UD, UK

Abstract. Key Performance Indicators (KPIs) are the main instruments of Business Performance Management. KPIs are the measures that are translated to both the strategy and the business process. These measures are often designed for an industry sector with the assumptions about business processes in organizations. However, the assumptions can be too incomplete to guarantee the required properties of KPIs. This raises the need to validate the properties of KPIs prior to their application to performance measurement.

This paper applies the method called EXecutable Requirements Engineering Management and Evolution (EXTREME) for validation of the KPI definitions. EXTREME semantically relates the goal modeling, conceptual modeling and protocol modeling techniques into one methodology. The synchronous composition built into protocol modeling enables traceability of goals in protocol models and constructive definitions of a KPI. The application of the method clarifies the meaning of KPI properties and procedures of their assessment and validation.

1 Introduction

Key Performance Indicators are cumulative measures of system achievements during a given time period. The achievements and the corresponding KPIs are related to system goals. The importance of the correct design of KPIs is clarified by their twofold nature. Namely, KPIs are calculated from the operational data, but they are interpreted at the strategic or tactical levels and often used by authorities to make decisions about the payment for the fulfilled work.

KPIs are usually designed for an industry sector with the assumptions made about the business processes in organizations. However, the business process in individual organizations may deviate from the business process used for the KPI definition. Moreover, the values of KPIs are often derived from the information about several businesses of different sectors. Incomplete assumptions about the business processes used for KPI definitions may result in different interpretation of KPIs. The organizations may become incompatible with respect to KPIs. Incomplete assumptions may leave the room for manipulation of KPI values to achieve better report numbers. The management science indicate this situation as "unreliable" and "plan oriented" KPIs [8].

B. Shishkov (Ed.): BMSD 2013, LNBIP 173, pp. 128–149, 2014.

Therefore, the definitions of KPIs and the completeness of assumptions about the underling processes should be validated. Because of the above mentioned twofold nature of KPIs, validation of their definitions demands both the operational models and the strategic models. The operational models are the executable process models for collecting the cumulative measures during model execution. The strategic models are the goal models for reasoning on KPI properties. The approach for validation of KPI definitions should use the related semantics for the goal models and the process models. The approach should enable building simple and easy changeable executable models. The changeability is needed to correct the assumptions about the business process used for KPI definitions.

In this paper, we propose to use the method called EXecutable Requirements Engineering Management and Evolution (EXTREME) [17] for validation of KPI definitions.

EXTREME is a combination of goal modeling, conceptual modeling and protocol modeling [17]. The synchronous composition built into protocol modeling enables the traceability of goals and concepts in the components of the protocol model and the interpretation of KPIs both in terms of goals and in terms of processes. The execution of the protocol model with the upturn and downgrade business data is used to validate whether the KPIs indicate the upturn and downgrade tendency, and whether the values of KPIs can be manipulated.

We build our work upon the existing methods [16,19,9] presented in Section 2 and use the conceptual basis of other methods. We have found that none of these methods can support the easy changeable executable process models needed for validation of properties of KPIs. We assume that the main reason for that is the semantic incompatibility of the goal modelling approaches and conventional process modelling approaches indicated in [9].

Section 3 describes the EXTREME method that exploits the semantic compatibility of the goal modelling and protocol modelling approaches. The semantics of the protocol modelling approach is described with the emphasis on the model execution that is needed for definition of KPIs.

Section 4 presents the formalization of the definition of KPIs from the protocol model point of view and interpretation of properties of KPIs in terms of this definition.

Section 5 describes the case for validation of KPIs presented in the program "Improving Access to Psychological Therapies (IAPT)" [5]. It reports the results of application the EXTREME method for the case. A triple of the goal, conceptual and executable protocol models is built on the basis of the IAPT document. We discuss the results of validation of properties of KPIs.

Section 6 presents conclusions and future work.

2 Related Work

2.1 Approaches for KPI Modelling

– A Performance Indicator (PI) is formalized in [16] as a concept with a number of attributes: Name, Definition, Type (continuous or discrete), Time

Frame, Scale, Min Value, Max Value, Source (Law, Company policy, Mission Statement), Owner ("the performance of which role or agent does it measure"), Threshold ("the cut-off value separating changes in the value of the performance indicator considered small and changes considered big"), Hardness ("a performance indicator can be soft or hard where soft means not directly measurable, qualitative, e.g. customers satisfaction") [16].
In order to find the values of the attributes, the authors rely on documents, expert knowledge and previous conceptual models. They indicate that it is not easy to find the information about all proposed attributes in the documentation.
The second concept used for the PI formalization is the performance indicator expression. It is "a mathematical statement over a performance indicator evaluated to a numerical, qualitative or Boolean value for a time point, for the organization, unit or agent." [16]. For example, $Response\ Time \leq 48hours$. The authors suggest to specify the required values of PIs as constraints estimated by experts with respect to a goal. The relations between different PIs are also modelled using the performance indicator expressions.
Let us notice that this formalization does not answer the question how a KPI can be calculated. The authors claim that they need to integrate the performance view with the process, organization and agent-oriented views of the real organizations. However, there is no information about the process semantics used for modelling and no evidence about validation of the PIs using processes. The authors do not involve the process view in the formalization of an indicator. In any case, the authors write about the process views of real organizations, whereas it is often needed to validate the PIs before their implementation in organizations.

– MetricM [19] is another method formalising KPIs. It "is built upon and extends an enterprise modeling approach to benefit from the reuse of modeling concepts and provide relevant organizational context, including business objectives, organizational roles and responsibilities."
The modelling language MetricML used in MetricM "adds essential concepts to modeling performance indicators..." The concept *Indicator* is used to present a KPI. The MetricML *Indicator* metatype is used for modeling its relations to other indicator types, to reference object types representing organizational context and to goal types.
An alternative "attribute" approach, used by MetricM, conceptualizes performance indicator as a (meta-) attribute of metatypes: e.g. "average throughput time" of a business process type or "average number of employees" of an organizational unit type. We partially use these alternative approach for our formalization in the next section. However, MetricM uses declarative models of performance indicators. The models of underlying processes, needed for execution and validation of KPI properties, are not used in MetricM.

The general tendency of two approaches, presented above, is to postpone the validation of the KPI properties to the moment when the process model of the organization is ready.

In this paper, we claim that the early validation of KPI properties on a business process, used for the KPI definition, may eliminate incompleteness of assumptions about the business process and prevent application of unreliable KPIs.

2.2 Goal Modelling Approaches and Modelling of KPIs

All existing approaches agree that the KPIs relate the system goals and processes. However, the process models are not used for validation of KPIs. Let us look what are the reasons for that.

There are goal oriented approaches, such Knowledge Acquisition in autOmated Specification (KAOS) [3], the User Requirements Notation (URN) [6] and i* modelling framework [20]. These approaches relate goals, business concepts and business processes. KAOS applies state machines to model behaviour of concepts. The URN applies a scenario modelling notation called Use Case Maps (UCM) [2]. Both approaches experience problems caused by the semantic incompatibility between the goal models and process models.

Letier at al [9] explain that the synchronous temporal logic used for goal modelling is interpreted over sequences of states observed at a fixed time rate. On the other hand, the conventional process models (UML state machines, use case maps, UML activity diagrams [14], Coloured Petri Nets [7]) use asynchronous temporal logics that are interpreted over sequences of states observed after each occurrence of an event. Thus, the temporal logic operators have very different meanings in synchronous and asynchronous temporal logics.

The process models built in asynchronous approaches accept the recognised messages, events or operation calls even if the state of the model is not appropriate to handle them. In such states, the messages, events or operation calls are kept in queues, bags or buffers to be handled in an appropriate state of the model. As a result, the behaviour model contains many intermediate states that are not justified by goals and declarative requirements. Analysis of intermediate states may be relevant for validation of asynchronous implementation. However, the goals and the KPIs are defined at a different level of abstraction, namely at the tactical and strategic level, i.e. at the level of observable states of the system.

Letier at al. [9] admit that in order to be semantically equivalent to the synchronous KAOS models, the derived event-based behaviour models need to refer explicitly to timing events. In other words, the event-based models should have elements of synchronization.

3 EXTREME: Goal Modelling with Protocol Modelling

Protocol Modelling [12] is an event-based modelling approach with the elements of synchronization needed for relating the goal and process models. The EXTREME [17] method exploits the semantic compatibility of goal and protocol models in order to simplify executable requirements engineering.

The method combines goal modelling, conceptual modelling and protocol modelling into one method to collect all the information needed for reasoning. The goal modelling and conceptual modelling are similar to the KAOS

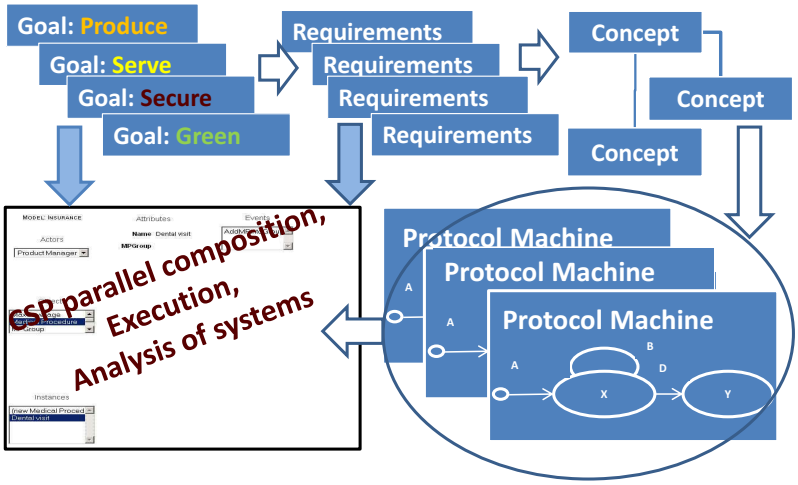

Fig. 1. The EXTREME method

method [3]. These steps are shown in Figure 1 as boxes named Goal, Requirement and Concept. Instead of the UML state machines and activity diagrams used by KAOS, EXTREME models the concept's behaviour as Protocol Machines (Figure 1) composed into a Protocol Models. The Protocol Model is executed using the Modelcope tool [11]. It is shown in Figure 1 as a screen for the user interface. The results of the execution are interpreted in terms of goals and requirements. The interpretation is indicated by the blue arrows.

Goals and concepts are modelled in the declarative way as snapshots of desired system behaviour. Protocol models use a form of synchronous CSP-parallel composition extended with data. They model only the quiescent states of the system that can be easily compared with the states specified for requirements and goals. The goal models and protocol models are semantically coherent. All states of protocol models can be interpreted in the goal semantics. This eases reasoning on models in terms of goals, goal refinement and identification of missing requirements [17].

Although there were many applications of the synchronous CSP parallel composition operator in the architecture description languages [1] and in programming languages [13], only after the extension of this operator for machines with data, made by A.McNeile [12], the operator became practical for business system modelling. The Protocol Modelling proposed in [12] enables coping with complexity of business modelling because the synchronous semantics decreases the data space of models. The process modelling with synchronous semantics solves the semantic mismatch between goal models and process models.

3.1 Protocol Models and Their Execution

Protocol Machine.

A building block of a protocol model is a protocol machine:

$$PM = (E, S_0, S, A, T),\ where$$

- $E = \{e_i\},\ (i = 1, ..., I;\ i, I \in N)$ is an alphabet of event types e_i, i.e. a non-empty finite set of recognized event types coming from the environment. An event type is a tuple of event attributes of different types: $e_i = (A_1^{ei}, ..., A_h^{ei});\ h \in N$. An instance of an event carries data. These data are used to update local storages of protocol machines. An attribute of type *Date* may be used to carry the time moment of event acceptance.
- S_0 is the initial state;
- $S = \{s_j\},\ (j = 1, ..., J;\ j, J \in N)$ is a non-empty finite set of states.
- $A = \{a_k\},\ (k = 1, ..., K;\ k, K \in N)$ is a finite set of attributes of different types. The set can be empty.
- $T = \{t_m\},\ (m = 1, ..., M;\ m, M \in N)$ is a finite set of transitions. $t_m = (s_x, e, s_y),\ s_x, s_y \in S, e \in E$. The set can be empty. The values of attributes are updated only as a result of a transition, i.e. as a result of event acceptance.

Synchronous composition.

In the initial state, a protocol model PM is a CSP parallel composition of protocol machines each of which presents a protocol machine type is state *new*. Initially, there are only the machine types serving as patterns for creating instances of protocol machines.

The instances are created by acceptance of events.

At any state, a system model PM is a CSP parallel composition of finite set of instances of protocol machines. There are multiple instances of each protocol machine type.

$$PM = \underset{i=1}{\overset{n}{\|}} PM_i = \underset{i=1}{\overset{n}{\|}} (E^{PM_i}, S_0^{PM_i}, S^{PM_i}, A^{PM_i}, T^{PM_i}) = (E, S_0, S, A, T),\ n \in N.$$

A Protocol Model PM remains a protocol machine, the set of states of which is the Cartesian product of states of all composed protocol machines [12]:

- $E = \bigcup_{i=1}^{n} E^{PM_i}$ is the set of events;
- $S_0 = \bigcup_{i=1}^{n} S_0^{PM_i}$ is the initial state;
- $S = \prod_{i=1}^{n} S^{PM_i}$ is the set of states;
- $A = \bigcup_{i=1}^{n} A^{PM_i}$ is the set of attributes.

The set of transitions T of the protocol model is defined by the rules of the CSP parallel composition [4]. The rules synchronise transitions T^{PM_i} of protocol machines. Namely, a Protocol Model handles only one event at a time. An event can be accepted only if all protocol machines having this event in their alphabets are in the state where they can accept this event. Otherwise the event is refused.

System Model Execution.

An execution of a protocol model of a system is a sequence of transitions. Processing of an accepted event is instantaneous: it does not take any time. A time moment of event processing may be assigned to the event and saved as a protocol machine attribute. The time moment of the event, that creates a protocol machine presenting a business object, is often useful for calculation of KPIs.

The initial state of an execution may contain any set of instances of protocol machines. At any moment, each protocol machine is situated in one of its states and its attributes have values of their types defined by the history of accepted events. Any state of any execution is quiescent, i.e. it does not change without an acceptance of a new event.

Dependent Protocol Machines. Derived states.

"Two machines having elements of their alphabets in common is not a source of dependency between them" [12]. The dependency means that one protocol machine needs to read the state of another machine to calculate its own state including the attributes. In any quiescent state, a function of this state can be calculated resulting in the extending of the state space of the protocol machine. The state space is extended by the state space of the dependent machines.

Protocol machines can read the state of each other, but cannot change it. This ability of protocol machines to read the state of each other will be used in the next section for the KPI modelling and calculation.

Observational Consistency.

As the CSP composition is applied to all instances of protocol machines in the executable model, it gives to the models the property called observational consistency [10]. This property means that a protocol machine may be added to and deleted from the model or locally changed. The trace behaviour of other protocol machines is not affected by the behaviour of added, deleted or modified protocol machines [10].

4 Definition of KPIs and Their Properties in EXTREME

We use the semantics of a protocol model (section 3.1) to give a constructive definition of a KPI. The definition should specify how a KPI can be derived from a protocol model.

Definition 1. *Let a system be presented as a protocol model PM (section 3.1)*

$$PM = \underset{i=1}{\overset{n}{\|}} PM_i,\ (n \in N) = \underset{i=1}{\overset{n}{\|}} (E^{PM_i}, S_0^{PM_i}, S^{PM_i}, A^{PM_i}, T^{PM_i}).$$

A KPI is a cumulative function of the cardinality of a set of selected business objects (presented by protocol machines) and the values of their attributes. The selected business objects give the value `true` *to the selection predicate* $\psi(A^{PM_j}, t, I, V_A)$. *The selection predicate compares the state and attribute values of each protocol machine with the border values of attributes* V_A, *the moment of selection* t *and the time interval* I.

$$KPI = f(\left|\bigcup PM_j\right|, \bigcup A^{PM_j}),\ where$$
$$PM_j : (\psi(A^{PM_j}, t, I, V_A) = true);$$
$$(0 \leq \left|\bigcup PM_j\right| \leq n).$$

An algorithm for calculation of a KPI includes

- *a predicate* $\psi(A^{PM_i}, t, I, V_A)$ *for the selection of a number of business objects (presented as protocol machines) using the time of calculation* t, *time interval* I *and a set of given border values of object attributes* V_A;
- *a cycle for selection and counting the number of business objects presenting the state of the model that meet the true value of selection predicate;*
- *a KPI calculation formula which arguments are the number of selected business objects and the cumulative variables depending on the attributes of selected business objects.*

The constructive definition of a KPI clarifies the definitions of its properties. We have chosen a set of properties proposed in [8,15] to formalise them on the basis of our definition and the system protocol model.

1. The first property demands that *a KPI should be in a quantifiable form.* Quantification is an act of selection and counting. Our definition of a KPI shows that selection and counting of instances of business objects (protocol machines) of a particular sort is the way of quantification. The result of selection and counting is used as an argument of the function for calculation of a KPI or for the access to the attribute values of selected objects.
2. The second property says: *A KPI needs to be sensitive to changes.* This property is about the design of the for a KPI calculation. Our model shows that the argument of the function can fall into two categories: a number of business objects or a variable that depends on values of attributes of the group of selected objects. The changes of the sensed elements in the model should cause the corresponding changes of the KPI. *The sensitivity is the minimum magnitude of the change of the sensed element of the model, required to produce a noticeable value of a KPI.* Constructing a KPIs function as a ratio or a cumulative function may affect the sensitivity.

3. The third desired property states that: *A KPI should be linear.* This simplifies the decision making.
   Our definition shows the possible arguments of the KPI formula. If the sensed argument in the model has been identified, it is the matter of the function choice to make the KPI function linear for this argument.
4. The fourth property is: *A KPI should be semantically reliable.*
   From the modelling perspective, we measure the semantic reliability of a KPI by the number of additional assumptions that need to be made in order to derive the KPI value. If the set of additional assumptions is empty, the KPI is semantically reliable. The next section presents an example of reliable and unreliable KPIs.
5. The fifth property relates the KPIs with goals of the system:
   A KPI should be oriented to improvement, not to conformance to plans.
   For validation of this property on a model, the improvement of the system should be defined from the definition of system goals. The improvement should be related to changes of a KPI in sequential time intervals. In order to test the changes of a KPI, several scenarios need to be executed and populate the model with objects. The model of the system and the system itself should guarantee that the arguments of the KPI formulas are objectively changed by the business process and cannot be manipulated. The executable model can help to identify the scenarios of system execution that lead to the KPI values reflecting both the improvement and the downgrade. The scenarios for manipulation of KPI values can be also identified. A manipulative scenario is a scenario of fraud. Existence of these manipulative scenarios is often caused by the incomplete assumptions about the processes for KPI derivation. It can also indicate the problems with business processes when the roles and the access rights are not specified.
   From the modelling perspective, we measure the improvement orientation of a KPI by the number of manipulative scenarios. If the set of manipulative scenarios is empty, the KPI is oriented to improvement.
   Our practical study illustrates all the properties and shows an example of a plan oriented KPI and an example of an improvement oriented KPI.

5 Case Study

The method for validation of KPIs is shown in Figure 2. The ovals show the input and the output. The boxes depict steps of the method, and the arrows indicate model refinement.

The input for the method application is a document that defines KPIs for a business sector. The KPIs are already designed, and some relevant concepts and steps of the business process are present in the definitions of KPIs. The brief summary of the document [5] is presented below.

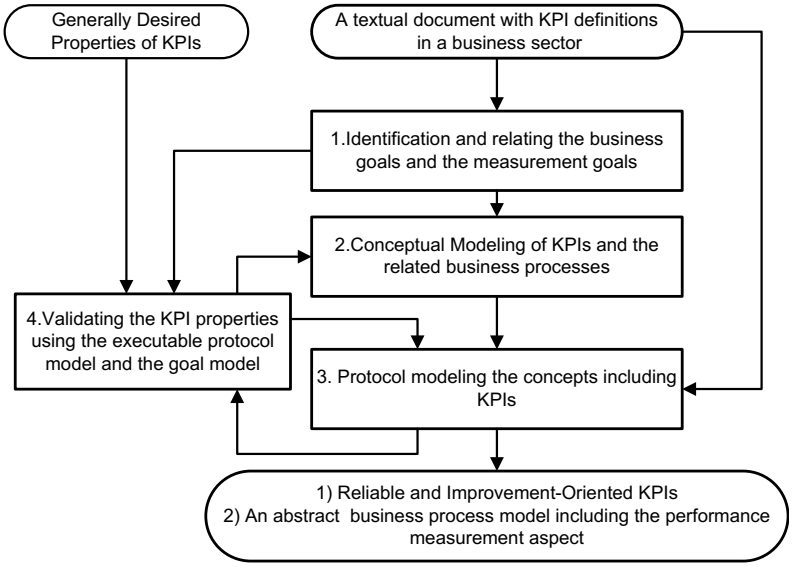

Fig. 2. Using EXTREME for validation of KPIs

KPIs of the Program for Improving Access to Psychological Therapies [5].

- KPI1: Level of Need. It presents the number of people who have depression and/or anxiety disorders in the general adult population. The number presenting population is produced as a result of the Psychiatric Morbidity Survey.
- KPI3a: The number of people who have been referred for psychological therapies during the reporting quarter.
- KPI3b: The number of active referrals who have waited more than 28 days from referral to first treatment/first therapeutic session (at the end of the reporting quarter).
- KPI4: The number of people who have entered psychological treatment, (i.e. had their first therapeutic session) during the reported quarter is related to the concept person.
- HI1: Access Rate. It indicates the rate of people entering treatment from those who need treatment $HI1 = \frac{KPI4}{KPI1}$.
- KPI5: The number of people completed treatment.
- KPI6: The number of people moving to recovery. This number sums up those who completed treatment, who at initial assessment achieve "Caseness" and at the final session - did not.
- KPI6b: The number of people who have completed treatment but were not at "Caseness" at initial assessment.
- HI2: Recovery Rate. It is calculated using the formula $HI2 = \frac{KPI6}{(KPI5 - KPI6b)}$.

The IAPT document does not provide information about KPI2 and KPI6a and states that they are no longer collected.

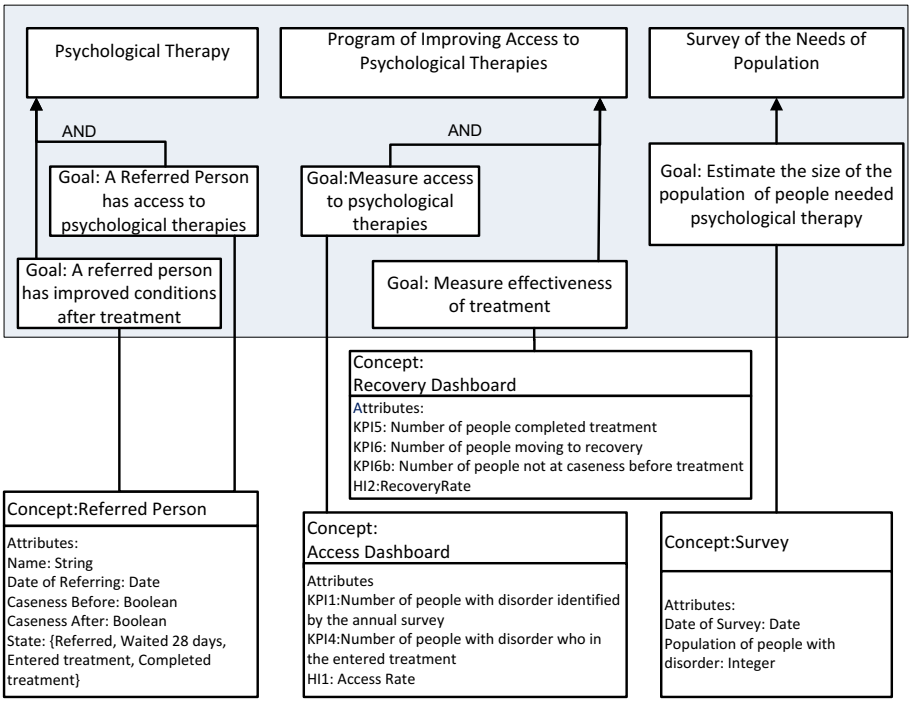

Fig. 3. Goals, Concepts and Protocol Models

The notion of "Caseness" is defined as a result of a condition assessment procedure. The procedure is applied to a referred person. There is no information about the rules of assessment and the values of "Caseness".

Two indicators are called High Indicators (HI). HIs are KPIs calculated from other indicators. In the terminology of Popova and Sharpanskykh [16], the IAPT KPIs can be called PIs and IAPT HIs can be called KPIs. We follow the terminology of the IAPT document [5] and call all indicators KPIs.

5.1 Identification and Relating the Business Goals and the Measurement Goals

In the IAPT document [5], we recognize the goals of measurement:

- *"Measure the access to the psychological therapies."*
- *"Measure the effectiveness of the psychological treatment."*

It is supposed that the underlying business processes:

- *"Estimate the size of population of people needing psychological therapy."*

and guarantee that

– *"A referred person has access to psychological therapy."*
– *"A referred person has improved conditions after treatment."*

The goals indicate three separate business processes: "*Survey of the Needs of Population*," "*Psychological therapy*" and "*Program for Improving Access to Psychological Therapies*".

The upper (grey) part of Figure 3 presents the goal model similar to the models built in Goal-Oriented methods [9]. The boxes are the goals and sub-goals. Goals are refined by the sub-goals that are combined in this case using the logical operator *AND*.

5.2 Conceptual Modelling of KPIs and the Related Processes in the Organization

As in other approaches [16,19], the goals of each process are refined to concepts with attributes. The information about the concepts is taken only from the IAPT document [5]. Concepts are depicted as boxes in the lower (white part) of Figure 3.

The concept *Survey* is the result of the process *Survey of the Needs of Population*.

The concept *Referred Person* is the subject of *Psychological Therapy* mentioned in the goals. We use a generic attribute *State* and identify its possible values of state from the IAPT document. For example, the names of the states of the life cycle of the *Referred Person* are *Referred*, *Waited 28 days*, *Entered treatment* and *Completed treatment*. The results of the condition assessment are modelled by two attributes *CasecessBefore* and *CasenessAfter*. As there is no indication about the type of *Caseness*, we assume that the type is *Boolean*.

In the search of the generic concepts for modeling of KPIs we decided to follow an approach suggested by Strecker et al [19]. We use a concept to present a family of measures for each goal of the measurement. We call such a concept a *Dashboard*. As in the business intelligence, an instance of a *Dashboard* presents a collection of values of measures supporting a particular request.

For example, an instance of the concept *Access Dashboard* shows the current values of indicators *KPI1, KPI3a, KPI3b KPI4* and *HI1*, measuring the access to the therapies. An instance of the concept *Recovery Dashboard* shows the values of the recovery indicators.

The concepts look like UML classes. However, the scarce information from the IAPT document does not allow us to build a complete class diagram and assign roles and relations.

5.3 Protocol Modeling of the Concepts, Including KPIs

In EXTREME, the concepts are modelled as protocol machines (defined in section 3.1).

The Concept *Survey*. is modelled as a protocol machine `Survey`. The protocol model of the `Survey` is described as follows (Figure 4):

```
OBJECT Survey
NAME SurveyName
ATTRIBUTES  SurveyName: String,
            Population:Integer,
            DateOfSurvey:Date
STATES created
TRANSITIONS @new*CreateSurvey=created

EVENT CreateSurvey
ATTRIBUTES  Survey:Survey,
            SurveyName:String,
            Population:Integer,
            DateOfSurvey:Date
```

The metacode, presented above, shows that a protocol machine is a state-transition system. It has its local state described using the keyword `STATES` and `ATTRIBUTES`. A transition from the initial state `@new` is triggered by event `CreateSurvey` which carries data of types `Survey:Survey`, `SurveyName:String`, `Population:Integer`, `DateOfSurvey:Date`.

Each instance of the `Survey` is created by accepting an event `CreateSurvey`. The acceptance of an event `CreateSurvey` brings with its attribute `Population` the number of people who have depression and(or) anxiety disorders and with its attribute `DateOfSurvey` the value of the attribute of the protocol machine `Survey`. Only the `Survey` in state "`created`" can provide the values of its attributes of the `LevelOfNeed` and `Population` for performance indicators.

The Concept *Referred Person*. The set of transitions and the state space of a protocol machine can be split into behaviours for the sake of separation of concerns. For example, the concept *Referred Person* is presented as the protocol machine `Referred Person` that `INCLUDES` behaviours `Treatment` and `Assessment`.

Attributes `CasenessBefore:Boolean` and `CasenessAfter:Boolean` store the results of assessment of the patient's conditions.

```
OBJECT ReferredPerson
    NAME PersonName
INCLUDES Treatment, Assessment
        ATTRIBUTES  PersonName: String, DateOfReferring:Date,
        STATES  referred,  28daysWaited,left
        TRANSITIONS @new*Refer=referred,
                    referred*Leave=left,
                    referred*Wait=28daysWaited,
                    left*Return=referred,
                    28daysWaited*EnterTreatment=28daysWaited
BEHAVIOUR  Treatment
    ATTRIBUTES CasenessBefore:Boolean,
        CasenessAfter:Boolean,
            DateOfCompletion:Date
            STATES   entered, completed,
    TRANSITIONS @new*EnterTreatment=entered,
```

```
                entered*CompleteTreatment=completed
BEHAVIOUR Assessment
    STATES   assessedBefore, assessedAfter
    TRANSITIONS @new*AssessBefore=assessedBefore,
                assessedBefore*AssessAfter=assessedAfter
EVENT Refer
    ATTRIBUTES ReferredPerson:ReferredPerson, PersonName:String,
            DateOfReferring:Date
EVENT Leave
    ATTRIBUTES  ReferredPerson:ReferredPerson
EVENT Wait
    ATTRIBUTES  ReferredPerson:ReferredPerson
EVENT Return
    ATTRIBUTES  ReferredPerson:ReferredPerson
EVENT EnterTreatment
    ATTRIBUTES ReferredPerson:ReferredPerson,CasenessBefore:Boolean
GENERIC AssessBefore
    MATCHES EnterTreatment
```

Figure 4 shows the protocol machines graphically. Protocol machines look like state machines. However, they have different semantics.

1) The `INCLUDES` relation of protocol machines is shown in Figure 4 as an arrow with a half-dashed end. The `INCLUDES` relation means that for every instance of `Referred Person` the instances of the dependent protocol machines `Treatment` and `Assessment` are created. The behaviours `Treatment` and `Assessment` are equally CSP parallel composed with other protocol machines.

The state space of a `Referred Person` is the Cartesian product of the state spaces of the `Referred Person` and the included behaviours `Treatment` and `Assessment`.

2) Protocol Modelling uses events as elements of interaction between the system and the environment and synchronization of protocol machines. Events are presented as data structures and can carry information. Each transition is labelled with an external event.

An event carries data that are used to update the attributes of an instance of the `Referred Person`. For example, the value of the `DateOfReferring` is entered with event `Refer`. `CasenessBefore` is updated with event `AssessBefore`. `CasenessAfter` is updated with event `AssessAfter`.

3) Protocol machines representing different levels of abstraction are easily composed using event matching mechanism.

For example, event `EnterTreatment` is matched with (considered as) event `AssessBefore` in the behaviour `Assessment`. Event `CompleteTreatment` is considered as `AssessAfter` in the behaviour `Assessment`. This is modeled using the keyword `GENERIC`.[1]

[1] In terminology of Aspect-oriented modelling events can be seen as join points. [10]

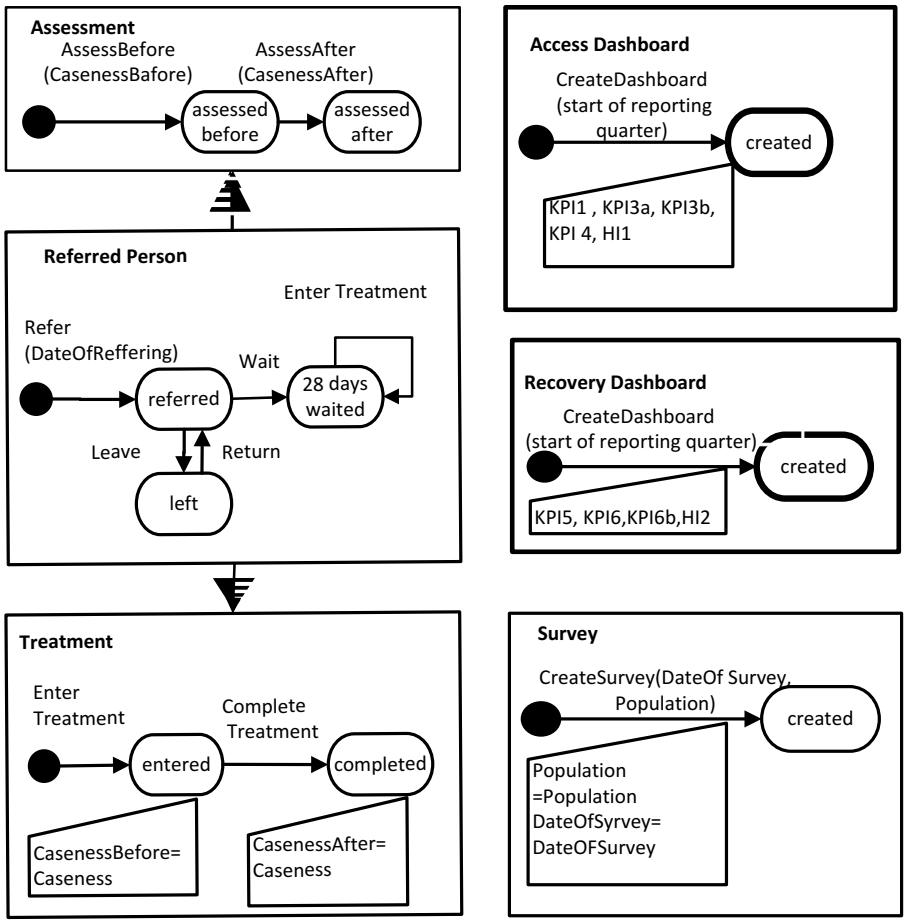

Fig. 4. Protocol Model

4) By submitting events a protocol model is deterministically populated with any number of instances of protocol machines.

5) As a consequence of the CSP parallel composition of protocol machines, the model has only the quiescent states, i.e. the states where the system does not proceed any event. All states can be justified by the system goals. Modelling and reasoning can be focused on the business semantics. Protocol machines in quiescent states can be selected for KPI measurement.

Dashboards and KPIs. The concept *Access Dashboard* and *Recovery Dashboard* are modelled as protocol machines.

```
OBJECT DashboardAccess
NAME  DashboardName
ATTRIBUTES  DashboardName:String,
            StartOfReportingQuarter:Date,
            !LevelOfNeed:Integer,
            !NumberReferredPersons:Integer,
            !NumberReferredPersonsWaited:Integer,
            !NumberOfEnteredTreatment:Integer,
            !AccessRate: Integer,
STATES created
TRANSITIONS
            @new*CreateDashboardAccess=created

OBJECT DashboardRecovery
NAME  DashboardName
ATTRIBUTES  DashboardName:String,
            StartOfReportingQuarter:Date,
            !NumberOfCompletedTreatment:Integer,
            !NumberOfPeopleMovingToRecovery:Integer,
            !NumberOfCasenessPeopleBeforeTreatment:Integer,
            !RecoveryRate:Integer
STATES created
TRANSITIONS
            @new*CreateDashboardRecovery=created

EVENT CreateDashboardAccess
ATTRIBUTES  DashboardName:String,
            DashboardAccess:DashboardAccess,
            StartOfReportingQuarter:Date

EVENT CreateDashboardRecovery
ATTRIBUTES  DashboardName:String,
            DashboardRecovery:DashboardRecovery,
            StartOfReportingQuarter:Date
```

The protocol machines `AccessDashbord` and `RecoveryDashboard` combine two groups of KPIs to monitor the access and recovery. Each dashboard protocol machine reads the state of protocol machines `Survey` and `Referred Person` and derives the values of own attributes presenting KPIs. The derived attributes of dashboard protocol machines, marked by the exclamation symbol "!", represent the corresponding KPIs.

The graphical representation (Figure 4) does not provide all the elements of the model. The complete protocol model of each of concepts *Access Dashboard* and *Recovery Dashboard* include algorithms (or state functions) for calculation of KPIs.

For example, the KPIs of the *Recovery Dashboard* are derived below. Each of the KPIs contains a part selecting objects (*selectInState()*). The selected objects are filtered on the bases of the selection predicate (*if* construction). The filtered objects are counted calculating the value of the corresponding KPI.

```
public class DashboardRecovery extends Behaviour {

public Date getStartOfReportingQuarter(){
    Calendar cal=Calendar.getInstance();
    Date StartQuarter = this. getDate("StartOfReportingQuarter");
    cal.setTime(StartQuarter);
return StartOfReportingQuarter;
}
public Date getEndQuarter(){
    Calendar cal=Calendar.getInstance();
    Date StartQuarter = this. getDate("StartOfReportingQuarter");
    cal.add(Calendar.MONTH, 3);
    Date EndOfReportingQuarter=cal.getTime();;
return EndOfReportingQuarter;
}

// KPI 5 Number of People Completed Treatment in the reported quarter
public int  getNumberOfCompletedTreatment() {
    int NumberOfCompletedTreatment=0;
        Date StartRQ = getStartQuarter();
        Date EndRQ=getEndQuarter();

Instance[] completedTreatment =   selectInState("ReferredPerson", "referred");
    for (int i = 0; i < completedTreatment.length; i++) {
    Date completionDate=completedTreatment[i].getDate("DateOfCompletion");
    String treatmentState = completedTreatment[i].getState("Treatment") ;
    if (completionDate.compareTo(StartRQ)>=0 &&
                completionDate.compareTo(EndRQ)<=0 &&
                treatmentState.equals("completed"))
        NumberOfCompletedTreatment+=1;
        }
    return NumberOfCompletedTreatment;
}

// KPI 6 Number of People Moving To Recovery
public int  getNumberOfPeopleMovingToRecovery() {
    int NumberOfPeopleMovingToRecovery=0;
        Date StartRQ = getStartOfReportingQuarter();
        Date EndRQ=getEndOfReportingQuarter();

Instance[] completedTreatment =   selectInState("ReferredPerson", "referred");
    for (int i = 0; i < completedTreatment.length; i++){
        Date completionDate=completedTreatment[i].getDate("DateOfReferring");
        String treatmentState = completedTreatment[i].getState("Treatment");
        Boolean CA=completedTreatment[i].getBoolean("CasenessAfter");
        Boolean CB=completedTreatment[i].getBoolean("CasenessBefore");
        if (completionDate.compareTo(StartRQ)>=0 &&
                completionDate.compareTo(EndRQ)<=0 &&
                treatmentState.equals("completed") &&
                CB==false && CA==true )
        NumberOfPeopleMovingToRecovery+=1;
        }
return NumberOfPeopleMovingToRecovery;
    }

// KPI 6b NumberOfCasenessPeopleBeforeTreatment from in the reported quarter
public int  getNumberOfCasenessPeopleBeforeTreatment () {
    int NumberOfCasenessPeopleBeforeTreatment=0;
        Date StartRQ = getStartQuarter();
        Date EndRQ=getEndQuarter();

Instance[] completedTreatment = selectInState("ReferredPerson", "referred");
    for (int i = 0; i < completedTreatment.length; i++) {
        Date completionDate=completedTreatment[i].getDate("DateOfReferring");
        String treatmentState = completedTreatment[i].getState("Treatment");
        Boolean CB=completedTreatment[i].getBoolean("CasenessBefore");
        if (completionDate.compareTo(StartRQ)>=0 &&
            completionDate.compareTo(EndRQ)<=0 &&
```

```
            treatmentState.equals("completed") &&
            CB==true)
        NumberOfCasenessPeopleBeforeTreatment+=1;
            }
    return NumberOfCasenessPeopleBeforeTreatment;
        }
```

The strategic indicators combine the KPIs producing rates. For example, the recovery rate combines three indicators:

```
// HI2 Recovery Rate KPI6/(KPI5-KP6b )
public int  getRecoveryRate() {
    int zn=(this.getInteger("NumberOfCompletedTreatment")-
    this.getInteger("NumberOfCasenessPeopleBeforeTreatment"));
    int RecoveryRate=0;
    if (zn==0){ RecoveryRate=0;}
    else {
    RecoveryRate=(100*this.getInteger("NumberOfPeopleMovingToRecovery"))/zn;
    }
return  RecoveryRate;
}
}
```

The complete protocol model can be found in [18]. The Model is executable in the Modelscope tool [11]. The tool generates an interface from the model. The interface is used to submit events, create objects and observe the values of attributes, i.e test the model and validate KPIs.

5.4 Validating the KPI Properties by Using the Executable Protocol Model, Goal and Conceptual Models

Let us analyze if the KPIs in our case study have the desired properties, mentioned in section 4.

1. *The quantifiability of KPIs.*

The KPIs have been modelled as attributes of the concept Dashboard. The protocol model has an algorithm for derivation for each KPI. The algorithm for calculation of a KPI is in the quantifiable form. It contains a predicate for the selection of a number of business objects (presented as protocol machines) using the given time interval and a set of given values of object attributes.

Protocol modelling has predefined select functions. Function

$$selectInState(\text{"}BehaviourName\text{"}, \text{"}State\text{"})$$

returns an array of objects (*"Behaviour name"*) , all of which are in the specified state (*"State"*). Function

$$selectByRef(\text{"}BehaviourName\text{"}, \text{"}AttributeName\text{"})$$

returns an array of objects, all of which have the specified attribute. The select functions enable modelling of quantifiable KPIs.

For example, the *KPI 6b* (the derive function has been shown in the previous section) selects *Referred Persons* initially assessed as Caseness: *CB=true* and then completed treatment within the reporting quarter *RQ*:

$$selectInState(\text{"}ReferredPerson\text{"}, \text{"}referred\text{"}) \ \&\ treatmentState.equals(\text{"}completed\text{"}) \ \&$$

$$completionDate.compareTo(StartRQ) \geq 0 \ \&\ completionDate.compareTo(EndRQ) \leq 0.$$

The number of selected referred persons is counted and gives the value of the indicator. All other KPIs of this case study are similar to *KPI6b.*

2. *The linearity of a KPI* can be tested using the executable model for each KPI. For testing, the model should be populated with objects. The properties of these objects should meet the selection criteria. For example, for the *KPI 6b* the model should be populated with referred persons initially assessed as caseness and completed treatment within the reporting period. The *KPI 6b* should count the amount of such referred persons. The population should also contain the objects that do not meet the selection criteria. Those objects should not be counted by the KPI derivation algorithm.

3. *The sensitivity* should be always validated for strategic indicators presented as rates. The large value of the denominator of a fraction can make the indicator insensitive. The sensitivity of the strategic indicators of our case *Recovery Rate* and *Access Rate* is increased using the percent scale.

4. *We measure the semantic reliability of a KPI* by the number of additional assumptions that need to be made in order to derive the KPI value. If the set of additional assumptions is empty, the KPI is semantically reliable. If the set of additional assumptions is not empty, then the execution and demonstration of the model to the users may be used to validate the assumptions because the users may have more knowledge than the KPI definition document.

For example, in our case, the procedure of assessment of the patient's conditions as *Caseness* is not specified by the IAPT document. We assume that a *Caseness* is a Boolean value coming from the environment. However, in practice, the *Caseness* may, for example, be assessed using a ten-point scale.Therefore, the KPIs, that depend on data assessed via *Caseness*, are not semantically reliable.

The KPI *HI:Recovery Rate* depends on the procedure of testing *Caseness* both before and after treatment:

$$HI2 : RecoveryRate = \frac{NumberOfPeopleMovingToRecovery}{(NumberOfPeopleCompletedTreatment - NumberOfCasenessPeopleBeforeTreatment)}.$$

We conclude that this KPI is not semantically reliable.

5. *The improvement orientation of a KPI* is validated for the strategic indicators as they are directly related to the goals of the system.

The meaning of improvement is rarely defined in the KPI documents. It demands extra efforts to guess the notion of improvement by analysing the goals of measurement.

There is a danger of replacing the improvement orientation of KPIs with the plan orientation. In such a case, the "desired" value of KPIs may be achieved through manipulating of numbers of instances in the business process.

Our case study presents examples of both an improvement-oriented KPI and a possibly plan-oriented KPI.

The KPI

$$HI1 : AccessRate = \frac{NumberOfEnteredTreatment}{LevelOfNeed}$$

is an example of an improvement-oriented KPI. It corresponds to the goal: *"A Referred Person has access to psychological therapies."* We assume that the improvement means the positive growth of the ratio of treated people to the people needed treatment.

Modelling shows that the numerator and the denominator of the KPI are the numbers derived from separate processes *Referred Person* and *Survey*. The processes are executed by different organizations, that do not depend upon each other. The *LevelOfNeed* comes from a *Survey*. The *NumberOfEnteredTreatment* is a sum of individually *Referred Persons*. The numbers of objects of separated processes grow independently through the model execution. The manipulation of the numerator and denominator of the KPI is unlikely. Only by the process improvement (for example, by shorten the time of waiting) the higher value of the *Access Rate* can be achieved. Therefore, we conclude that the KPI *HI1:Access Rate* is oriented to improvement.

The KPI

$$HI2 : RecoveryRate = \frac{NumberOfPeopleMovingToRecovery}{(NumberOfPeopleCompletedTreatment - NumberOfCasenessPeopleBeforeTreatment)}.$$

is an example of KPI that may become plan oriented and open to manipulations. For validation of the improvement orientation of this indicator, we use both the goals associated with KPIs and the model of the underlying process. The KPI corresponds to the goal *"A Referred Person after treatment has improved conditions"*.

A person moves to recovery if the assessment of Caseness before treatment is *false (sick)* and after treatment is *true (healthy)*. The improvement corresponds to the growth of the *Recovery Rate* of persons that move to recovery. If the procedures of the *Caseness* assessment and treatment are assigned to the employees of the same organization, who have their interest in high value of *Recovery Rate*, the value of *Recovery Rate* can be manipulated to meet the planned values. This can be done by assessing healthy people as sick before the treatment and sending them for the treatment and/or by assessing sick people as healthy after the treatment. The corresponding scenarios can be shown by model execution.

Validating this property, we conclude that the information in the KPI document is not sufficient to assure the improvement oriented *Recovery Rate*.

One of the possible solutions for improvement of the KPI document may be the following constraint to the business processes of organizations: "The assessment

of *Caseness* and treatment should be fulfilled by two independent institutions with different sources of financial support".

Another possible solution is the definition of the rules for *Caseness* assessment, including the assignment of organizations responsible for the treatment and assessment.

The presented examples show that the combination of the goal, conceptual and protocol models, used in EXTREME, provides a useful instrument for validation of KPI properties and leads to discovery of tacit constraints and rules in KPI definitions.

6 Conclusion

In this paper, we have proposed a method for modelling of KPIs and validation of their properties using the approach called EXecutable Requirements Engineering Management and Evolution (EXTREME).

The contribution of this paper is the constructive definition of a KPI. The definition specifies a procedure for calculation of a KPI in models and in information systems. The definition also clarifies the procedures for assessment and validation of properties of KPIs on models.

As a byproduct of the modelling of KPIs with the EXTREME method, an executable model of an abstract organization in an industry sector is produced. In future work, this model can be used for standardization of performance measures in an industry sector and for assessment of applicability of KPIs in real organizations.

References

1. Allen, R., Garlan, D.: Beyond Definition/ Use: Architectural Interconnection. In: Proceedings, Workshop on Interface Definition Languages, Portland, Oregon (1994)
2. Alsumait, A., Seffah, A., Radhakrishnan, T.: Use Case Maps: A Visual Notation for Scenario-Based Requirements. In: 10th International Conference on Human - Computer Interaction (2003), http://wwwswt.informatik.uni-rostock.de/deutsch/Veranstaltungen/HCI2003/
3. Dardenne, A., van Lamsweerde, A., Fickas, S.: Goal-directed requirements acquisition. Sci. Comput. Program 20(1-2), 3–50 (1993)
4. Hoare, C.: Communicating Sequential Processes. Prentice-Hall International (1985)
5. Improving Access to Psychological Therapies. IAPT Key Performance Indicator (KPI). Technical Guidance for Adult IAPT Services (2013), http://www.iapt.nhs.uk/silo/files/iapt-kpi-technical-guidance-201213-v20-.pdf
6. ITU. Formal description techniques (FDT). User Requirements Notation Recommendation Z.151 (11/08), http://www.itu.int/rec/T-REC-Z.151-200811-I/en
7. Jensen, K.: Coloured Petri Nets. Springer (1997)
8. Kueng, P.: Process performance measurement system - a tool to support process-based organizations. Total Quality Management 11(1), 67–85 (2000)

9. Letier, E., Kramer, J., Magee, J., Uchitel, S.: Deriving event-based transition systems from goal-oriented requirements models. Automated Software Engineering Archive 15(2), 175–206 (2008)
10. McNeile, A., Roubtsova, E.: CSP parallel composition of aspect models. In: Proceedings of the 2008 AOSD Workshop on Aspect-Oriented Modeling, AOM 2008, pp. 13–18. ACM, New York (2008)
11. McNeile, A., Simons, N.: (2005), http://www.metamaxim.com/
12. McNeile, A., Simons, N.: Protocol Modelling. A Modelling Approach that Supports Reusable Behavioural Abstractions. Software and System Modeling 5(1), 91–107 (2006)
13. Mount, S., Hammoudeh, M., Wilson, S., Newman, R.: CSP as a domain-specific language embedded in Python and Jython. In: Welch, et al. (eds.) [248], pp. 293–309 (2009)
14. OMG. Unified Modeling Language: Superstructure version 2.1.1 formal/2007-02-03 (2003)
15. Parmenter, D.: Key Performance Indicators, Developing, Implementing and Using Winning KPIs. John Wiley & Sons, New Jersey (2010)
16. Popova, V., Sharpanskykh, A.: Modeling organizational performance indicators. Information Systems 35(4), 505–527 (2010)
17. Roubtsova, E.: EXTREME: EXecuTable Requirements Engineering, Management and Evolution. In: Diaz, V.G. (ed.) Progressions and Innovations in Model-Driven Software Engineering, pp. 65–89. IGI Global (2013)
18. Roubtsova, E.: Protocol Model of the KPIs from the program "Improving Access to Psychological Therapies" (2013), http://www.open.ou.nl/elr/IAPT.zip
19. Strecker, S., Frank, U., Heise, D., Kattenstroth, H.: MetricM: A modelling method in support of the reflective design and use of performance measurement systems. Springer, Information Systems and e-Business Management 10, 241–276 (2012)
20. Yu, E.: Modelling Strategic Relationships for Process Reengineering. Ph.D. Thesis. Dept. of Computer Science, University of Toronto (1995)