A Method for Modeling of KPIs Enabling Validation of Their Properties

Ella Roubtsova
Open University of the Netherlands
6401DL,Valkenburgerweg 177
Heerlen, The Netherlands
ella.roubtsova@ou.nl

Vaughan Michell
Henley Business School
University of Reading
Whiteknight, Reading, RG6 6UD, UK
v.a.michell@henley.reading.ac.uk

ABSTRACT

Key performance indicators (KPIs) play an important role in making decisions for correct and timely investments. Validation of desired properties of KPIs demands the execution of business processes. However, KPIs are often designed for a business sector assuming fragmental knowledge about business processes. The validation of properties of KPIs is postponed until the KPIs application in business processes of organizations. This approach often results in misleading KPIs.

We present a method that enables validation of KPI properties without implementation of KPIs in organizations. The method takes advantages of a combination of goal modeling, conceptual modeling and executable Protocol Modeling. The compositional semantics of Protocol Modelling allows for building abstract executable protocol models using the fragmental information about business processes. The goal models and conceptual models are used for analysing the results of the process execution and reasoning about properties.

The method is demonstrated with a real KPI document from the medical sector.

Categories and Subject Descriptors

H.1.0 [**Information Systems**]: MODELS AND PRINCIPLES; D.2.1 [**Software Engineering**]: Software Architectures

General Terms

Design, Modelling

Keywords

Key Performance Indicators; Goal modeling; Conceptual modeling; Executable Protocol Modeling; Properties; Validation

1. INTRODUCTION

The triple known as "people, planet, profit" [4] has been chosen as a slogan by many modern businesses, trying to win the support of people and governments. The triple directs organizations to focus not only on the economic value, but also on the social and environmental value [4]. The new focus of organizations stimulates the search for the right measures of organizational success or key performance indicators (KPIs). The KPIs are used almost for any domain of our life, including medicine, education, services and green computing [10].

There is a large body of work on KPIs that reflects that KPIs are defined from the goals and strategies of organization types [5]. KPIs usually exist as families, which relate different management perspectives: strategic, tactical and operational, with the processes within the organization. The difficulties in definition of KPIs are caused by relating the abstract vocabulary at the strategic level and the vocabulary of structured sources of business processes. The KPIs are usually built to measure the dynamics toward business goals. The goals are related to a set of relevant processes. Often, the values of KPIs are derived from several processes.

The validation of properties of KPIs demands the execution of processes. However, the KPIs are developed for a business sector. Even businesses of the same sector have different variants of the business process. It is impossible to know all these variants of processes. Therefore, the KPIs are designed conceptually. Validation of properties is postponed to the moment of implementation of the KPIs in an organization. If the real process of an organization deviates from the assumptions used for KPI definitions, the KPIs may become inapplicable or misleading.

We show that the existing modelling approaches for KPI modelling [20; 18] do not construct executable processes at the stage of the KPI design and do not validate properties of KPIs.

As KPIs are defined at operational, tactical and strategic levels, their modelling demands an approach that easily composes processes at different levels.

Because KPIs are often derived from different processes, their modelling demands specific process modelling techniques with synchronization mechanisms.

In this paper, we propose to validate the KPI properties on a set of abstract executable processes before the implementation of KPIs in an organization. These abstract processes with KPI calculations can be later used for estimation the KPI applicability for organizations.

Our method is built upon existed methods for KPI modelling and on the Protocol Modeling technique. We illustrate our method with the case "Improving Access to Psychological Therapies (IAPT)" [7]. An abstract executable Protocol Model is built on the basis of the information from the documents with KPI definitions. The model is used for validation of the properties of KPIs.

The key contribution of this paper is investigating the possibility of using the Protocol modelling technique [14] for modelling of KPIs and evaluation of their properties. Another contribution is a method showing how to build protocol models for KPIs starting from documents describing KPIs and proceeding to goal, conceptual and protocol models. Because of the use of Protocol Modelling, our method compares favourably to other methods and provides capabilities for model execution and validation of properties at the early stages of KPI design.

We show the practical relevance of flexible Protocol Modelling for validation of KPI properties, as the violation of some properties suggests requirements and constraints on the abstract business processes of organizations that will use the KPIs.

The structure of the paper is the following.

In Section 2 we describe the properties of KPIs listed in literature. In Section 3, we show the shortcomings of existing approaches for modelling of KPIs and the advantages of using the Protocol Modelling approach. Section 4 describes a case study. In Section 5, we present our method for modelling of KPIs and validation of their properties. We illustrate our method with the case study. Section 6 presents future work and conclusions.

2. PROPERTIES OF KPIs

The properties of good KPIs are listed in the literature. Kechenham and Winchell [11] defined six desired properties separating KPIs from other measures. These properties may be seen as requirements for KPI validation.

We list the properties as they are formulated in [11] and discuss the usefulness of model execution for validation of each property.

1. *A KPI should be in a quantifiable form.*

Quantification means deriving a number or a conclusion from a set of instances of selected concepts in the models. Any KPI can be presented as a quantification predicate of the first-order logic [2] or as an algorithm selecting instances in particular state or with particular values of attributes. Validation of this property demands instantiation of conceptual models. Execution of the model is useful to ensure that the quantification expression derives the right numbers of instances from the model.

2. *A KPI needs to be sensitive to changes of the business process state.*

Any variation of the arguments in the KPI formula should be visible in the KPI value. The arguments are derived from the underlying business model. Wrong underlying model may introduce compensation of changes making them invisible. Validation of this property on an executable model ensures the clarity of the KPI definition. It ensures that the modeler is able to correctly understand the KPI application.

3. *A KPI should be linear.*

Linearity of a KPI means that there is a linear mathematical dependency between the value of the indicator and the values of data and state of the model. Model execution is needed to collect data and validate the KPI linearity.

4. *A KPI should be semantically reliable.*

Reliability means that the assumptions about the business process are valid for the process of organizations and the algorithms for KPI calculation use well defined procedures both for routine, as well as unexpected circumstances. Model execution is like testing of programs can clarify the assumptions.

5. *A KPI should be efficient.*

The KPIs should not duplicate each other.

6. *A KPI should be oriented to improvement, not to conformance to plans.*

This property relates the KPI definitions to strategic, tactical goals and to the underlying business processes. There is a danger that KPIs can be used to manipulate numbers instead of showing the improvement. The execution of the model can be used to predict scenarios of such a danger.

KPIs indicate how the system performs, i.e. behaves during its execution. Therefore, the properties of KPIs can be validated on the executable models of underlying processes.

3. RELATED WORK

3.1 Approaches for KPI modelling

Popova and Sharpanskykh [18] propose a formalization of the concept of a performance indicator (PI). The proposed formalization of a PI suggests a number of attributes: PI name, Type, Scale, Source, Owner, Threshold, Hardness. The authors indicate that it is not easy to find the information about all attributes in the documentation. The authors rely on documents, expert knowledge and previous conceptual models and do not involve the process view in formalization of the indicators.

The second concept used for 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. For example, $PI27 \leq 48h$." [18]. The authors suggest to specify the required values of PIs as constraints coming from goals.

The relations between PIs are modelled using predicates.

The authors claim that they integrate the performance view with the process, organization and agent-oriented views. However, there is no information about the process semantics used for modelling and no evidence about validation of the PI properties. In any case, the authors write about the process views of the real organizations, not about the abstract processes that we propose.

The method MetricM (Strecker et al [20]) "is built upon and extends an enterprise modeling approach to benefit from the reuse of modeling concepts to provide relevant organizational context, including business objectives, organizational roles and responsibilities." The method can be adapted to any enterprise modeling approach. The modelling language MetricML used in MetricM "adds essential concepts to modeling performance indicators and semantics to key modeling concepts." 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 conceptualizes performance indicator as (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 use this alternative approach for KPI modelling in our method.

MetricM uses declarative models. The model of underlying processes needed for validation of KPI properties are not used in MetricM.

The two approaches, presented above, build upon ideas of many earlier approaches to KPI modelling. The general tendency 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 preliminary validation of KPI properties is possible on an abstract process. Such early validation prevents the mistakes in the design of the process of the organization leading to misleading KPIs. The possibility to design an abstract process for KPI definition and then compose it with other sub-processes of an organization is offered by the semantic nature of Protocol Modelling.

3.2 Conventional Process Modelling and Protocol Modelling

Most of the conventional process modelling approaches have asynchronous semantics of handling interactions.

The approaches that are bundled in the UML (Activity diagrams, State machines, Sequence diagrams) [16], as well as Petri Nets [17] and Coloured Petri Nets [8] use one of asynchronous semantics. Some of these approaches, like Coloured Petri Nets and State Machines, support modelling with data. Other approaches, like sequence diagrams and activity diagrams, abstract from data and consider only interactions. However, the models built in all of these 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. This causes many intermediate states in the model that are not justified by goals and declarative requirements. Analysis of intermediate states may be relevant for validation of asynchronous implementation. However, 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 and the asynchronous modelling does not provide the right level of abstraction.

The synchronous modelling semantics is based on the CSP parallel composition operator defined by Hoare [6]. The operator defines that an event from environment is accepted by the model if all processes of this model are able to accept it. Otherwise, the event is refused.

Although there were many applications of the CSP parallel composition operator in the architecture description languages [1], in programming languages [15], only after the extension of this operator for machines with data, made by McNeile [14], the operator became practical for business system modelling. The Protocol Modelling proposed in [14] enables coping with complexity of business modelling. The reason is that the synchronous semantics decreases the data space of models.

Now we briefly describe the semantics of Protocol modelling [14].

Any protocol model consists of protocol machines.

Each protocol machine has an alphabet of recognised events. An event is described as a data structure.

A protocol model accepts one event at a time. An event is accepted only if all protocol machines, that recognise this event, are able to accept it. Otherwise, the event is refused. If an event is not recognised by the machine (does not belong to its definition) it is ignored. All the intermediate states of asynchronous models, caused by queues and arbitrary time of event arrival, as well as the states of event processing do not belong to the protocol model. The transition from one state to another happens only as a result of acceptance of an event.

A protocol machine has its local storage. The storage is updated only as a result of acceptance of an event. Instances of events contain data used to update local storages of protocol machines. The state of any protocol machine and its local storage can be read only as a result of event acceptance.

There are protocol machines with derived states that derive their state from the state of other protocol machines. Clearly synchronisation is the necessary condition of such a derivation.

As KPIs are usually calculated from the states of processes, the semantics of synchronous composition and state derivation provided by Protocol Modelling is an asset for KPI modelling.

As the CSP composition is applied to all available protocol machines, it gives to the models another valuable property. Protocol machines can be added and deleted from the model, and the trace behaviour of the other protocol machines will not be changed. This property is called observational consistency. The proof of it can be found in [12]. This property is useful for KPI modelling, as the KPIs are defined on abstract processes that have to be later composed with real processes in organizations. The application of this property for weaving the abstract processes used for the KPI design into the processes of organizations is out of the scope of this paper.

All these semantic advantages are the reasons of our choice to use Protocol Modelling in our method for modelling of KPIs.

4. CASE STUDY

We illustrate our method by applying it to the Program for Improving Access to Psychological Therapies (IAPT) [7]. The indicators need to measure a quarter on quarter improvement.

IAPT does not state the goals explicitly, but allows for identification of the following goals:

- *Measurement of the access to the psychological therapies*
  - 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 $HI1 = KPI4/KPI1$.

- *Measurement of the effectiveness of the psychological treatment*
  – 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.

    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".
  – 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 $HI2 = KPI6/(KPI5 - KPI6b)$.

The IAPT document says that the `KPI2` and `KPI6a` are no longer collected.

Two indicators are called High Indicators (HI). HIs are KPIs calculated from other KPIs.

In the terminology of Popova and Sharpanskyk [18], the IAPT KPIs of the IAPT can be called PIs and IAPT HIs can be called KPIs. We follow the IAPT document [7].

5. MODELLING OF KPIs AND VALIDATION OF THEIR PROPERTIES

The input of our method is a document that defines KPIs for a sector of organizations. The KPIs are already designed, and some relevant concepts and steps of the business process are present in the definition of KPIs.

The goal of our method is building an abstract process used for the KPI definitions and validating the KPI properties on it. We see this abstract process as an aspect of the business process of organization. If the KPI properties are not valid, the goal is to propose the constraints for the abstract process.

Our method consists of several steps producing related models at different levels of abstraction. The steps of our method are graphically presented in Figure 1. The ovals show the input and the output. The boxes depict steps of the method, and the arrows indicate model refinement.

5.1 Relating the goal of the KPIs measurement to the goals of the organizations in the sector

In the IAPT document [7], we recognize the goals of measurement:

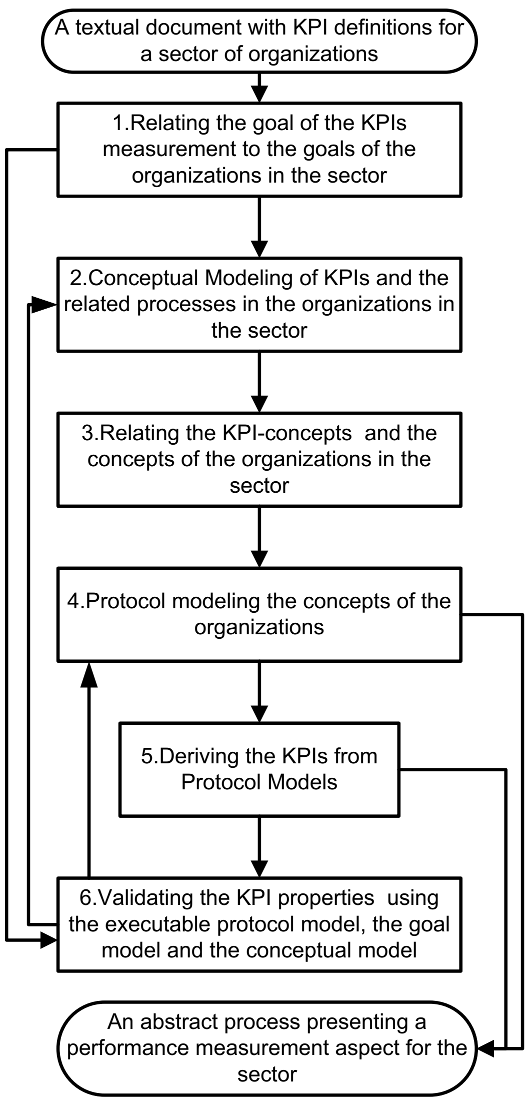
A textual document with KPI definitions for a sector of organizations
1.Relating the goal of the KPIs measurement to the goals of the organizations in the sector
2.Conceptual Modeling of KPIs and the related processes in the organizations in the sector
3.Relating the KPI-concepts and the concepts of the organizations in the sector
4.Protocol modeling the concepts of the organizations
5.Deriving the KPIs from Protocol Models
6.Validating the KPI properties using the executable protocol model, the goal model and the conceptual model
An abstract process presenting a performance measurement aspect for the sector

Figure 1: The method for modeling of KPIs enabling validation of their properties

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

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

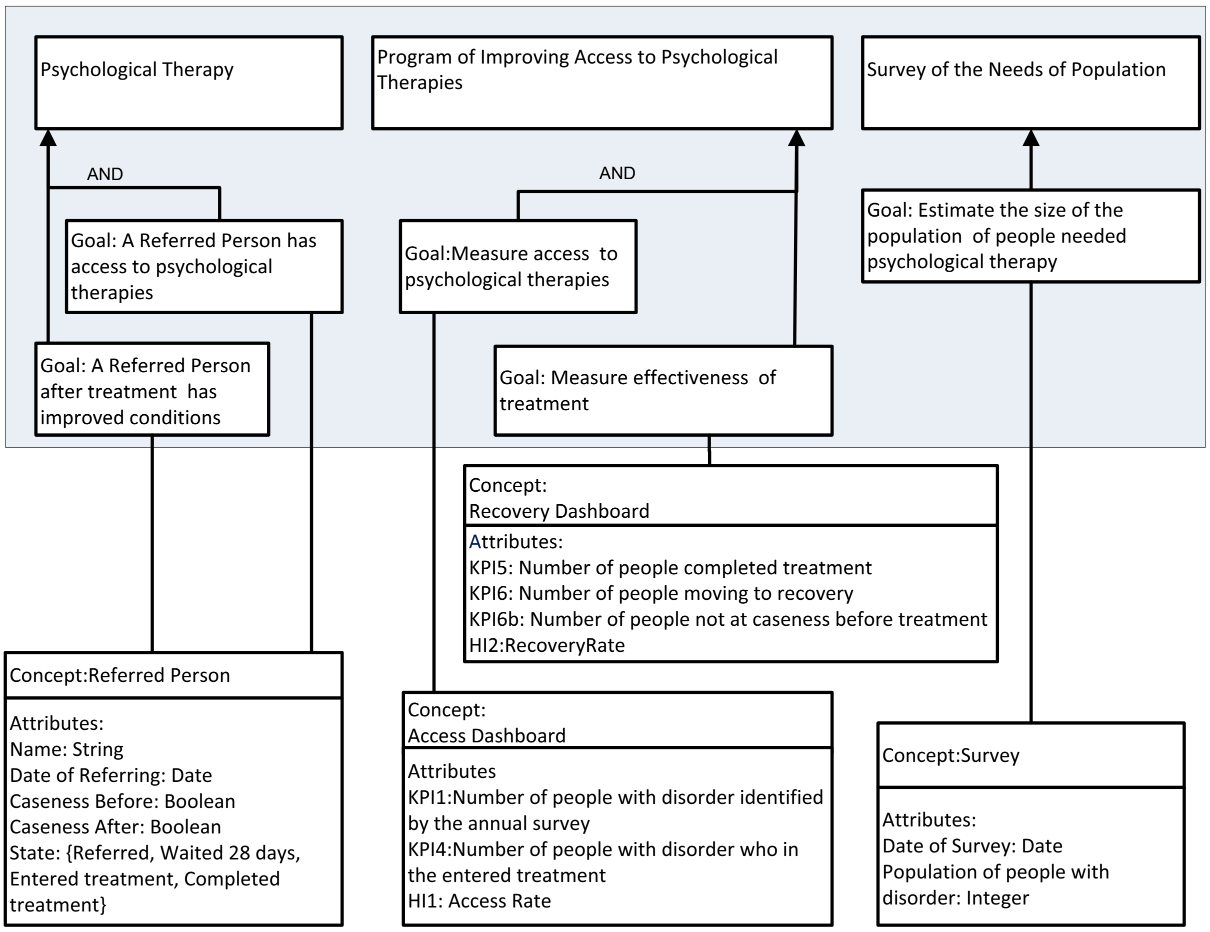
Psychological Therapy
Program of Improving Access to Psychological Therapies
Survey of the Needs of Population
AND
AND
Goal: A Referred Person has access to psychological therapies
Goal:Measure access to psychological therapies
Goal: Estimate the size of the population of people needed psychological therapy
Goal: A Referred Person after treatment has improved conditions
Goal: Measure effectiveness of treatment
Concept: Recovery Dashboard
Attributes:
KPI5: Number of people completed treatment
KPI6: Number of people moving to recovery
KPI6b: Number of people not at caseness before treatment
HI2:RecoveryRate
Concept:Referred Person
Attributes:
Name: String
Date of Referring: Date
Caseness Before: Boolean
Caseness After: Boolean
State: {Referred, Waited 28 days, Entered treatment, Completed treatment}
Concept: Access Dashboard
Attributes
KPI1:Number of people with disorder identified by the annual survey
KPI4:Number of people with disorder who in the entered treatment
HI1: Access Rate
Concept:Survey
Attributes:
Date of Survey: Date
Population of people with disorder: Integer

Figure 2: Goals, Concepts and Protocol Models

The upper (grey) part of Figure 2 presents the goal model similar to the models built in Goal-Oriented methods, for example [3]. The boxes are the goals and sub-goals. Goals are refined by 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 [18; 20], the goals of each process are refined to concepts with attributes. The information about the concepts is taken only from the IAPT document [7].

Concepts are depicted as boxes in the lower (white part) of Figure 2.

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*.

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

In the search of the generic concepts for modeling of KPIs we decided to follow an approach suggested by Strecker et al [20]. We use a concept to present the family of measures for each goal of the measurement. We call such a concept a *Dashboard*. As in the business intelligence domain, an instance of *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 access. An instance of the concept *Recovery Dashboard* shows the values of 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. Further, we need to have good arguments to make suggestions for document improvement. That is why we try to model KPIs using the IAPT document and analyze KPI properties. If the KPI properties cannot be guaranteed, this indicates the need for the document improvement. Moreover, the detailed suggestions for the document improvement may be motivated by desired properties.

5.3 Relating the KPIs to the business concepts

The attributes of the dashboard concepts, designed for modeling of KPIs, need to be derived from the concepts of the relevant business processes.

From the IAPT document, we extracted two relevant business processes and two corresponding business concepts for calculation of the KPIs of the IAPT program: *Survey* and *Referred Person*. Each of these concepts has its own attributes.

Conceptual modelling forces us to think of the needed attributes for KPI calculation. For example, the definition of KPIs says that the monitoring takes place quarterly. This implies that the concepts of the underlying process need the attributes representing the date of their appearance. The *Survey* gets its attribute *DateOfSurvey* and *Referred Person* gets its attribute *DateOfReferring*.

As the day of monitoring is not fixed, we also need a synchronization means that will trigger the derivation of the KPI values combined in the concepts *Access Dashboard* and *Recovery Dashboard*. We will show in section 5.5 that the semantics of derived states accurately supports the derivation of the KPI values.

5.4 Protocol modelling of the business concepts

Further we model concepts as protocol machines.

For instance, the concept `Survey` is modelled as a protocol machine `Survey`. The protocol model of the `Survey` is described as follows (Figure 3):

```
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
```

As we can see in the metacode, the 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 `Create-Survey` 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 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 *Refereed 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,
            enteredTreatment,
            completedTreatment
TRANSITIONS @new*Refer =referred,
referred*Decline=left,
left*Return=referred,
referred*Wait=28DaysWaited,
28DaysWaited*Leave=left,
28DaysWaited*EnterTreatment=28DaysWaited

BEHAVIOUR  Treatment
ATTRIBUTES CasenessBefore:Boolean,
           CasenessAfter:Boolean
STATES enteredTreatment,
   completedTreatment,
       left
TRANSITIONS
    @new*EnterTreatment=enteredTreatment,
    enteredTreatment*Leave=left,
    enteredTreatment*CompleteTreatment=
 completedTreatment

BEHAVIOUR  Assessment
STATES AssessedBefore,
       AssessedAfter
TRANSITIONS
    @new*AssessBefore=AssessedBefore,
AssessedBefore*AssessAfter=AssessedAfter

EVENT Refer
ATTRIBUTES  ReferredPerson:ReferredPerson,
            PersonName:String,
            DateOfReferring:Date,
EVENT Decline
ATTRIBUTES  ReferredPerson:ReferredPerson

EVENT Return
ATTRIBUTES  ReferredPerson:ReferredPerson

EVENT Wait
ATTRIBUTES  ReferredPerson:ReferredPerson

EVENT Leave
ATTRIBUTES  ReferredPerson:ReferredPerson

EVENT EnterTreatment
ATTRIBUTES ReferredPerson:ReferredPerson,
            CasenessBefore:Boolean

GENERIC AssessBefore
MATCHES EnterTreatment

EVENT CompleteTreatment
ATTRIBUTES ReferredPerson:ReferredPerson,
            CasenessAfter:Boolean

GENERIC AssessAfter
MATCHES CompleteTreatment
```

Figure 3 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 3 as an arrow with a half-dashed end. The `INCLUDES` relation means that for every instance of `Referred Person`

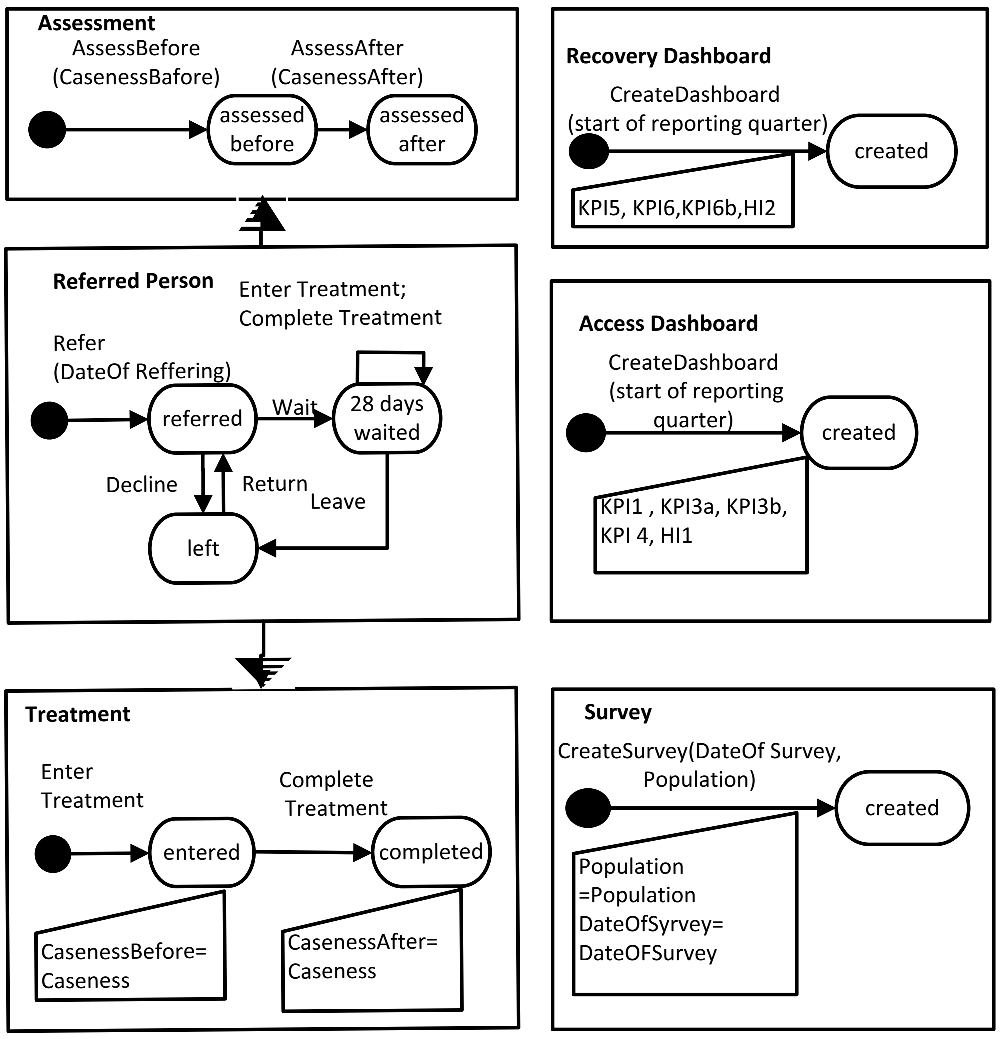

Figure 3: Protocol Model

an instance of `Treatment` and an instance of `Assessment` are created. The behaviours `Treatment` and `Assessment` are equally CSP parallel composed with other protocol machines.

The state space of `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.

As events are the structures that carry data, they allow us to update the attributes in the life cycle of an instance of the `Referred Person`. 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]

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. Quiescent states can be selected for KPI measurement. Protocol modelling has predefined select functions useful for definition of tactical KPIs.

[1]In terminology of Aspect-oriented modelling events can be seen as join points. [12]

5.5 Deriving the KPIs from the protocol models of concepts

The concepts *Access Dashboard* and *Recovery Dashboard* are modelled as protocol machines as follows:

```
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` present the KPIs monitoring the access and recovery by selecting and counting the instances of the protocol machines.

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 KPIs.

The graphical representation (Figure 3) does not provide all the elements of the model. The complete protocol model of the program for IAPT is presented with its metacode and small state functions used for deriving states. For example, the state of *Access Dashboard* is derived as follows. `AccessDashboard` reads the state of protocol machine `Survey` (but does not change it). It finds the `Survey` instance with the closest date and takes the value of the `Population` of this `Survey` and assigns it to own attribute `LevelofNeed`. The search function `selectInState("Survey", "@any")` selects the set of surveys to choose the latest survey from this set. The state function is presented below.

```
public class AccessDashboard extends Behaviour{
//KPI 1 Level of Need

public int  getLevelOfNeed() {
int LevelOfNeed=0;

// choose the date three years ago
Calendar cal = Calendar.getInstance();
cal.add(Calendar.YEAR, -3);
Date dd = cal.getTime();

Date Dashd =
this. getDate("StartOfReportingQuarter");

Instance[] existingSurvey =
selectInState("Survey", "@any");

for
(int i = 0; i < existingSurvey.length; i++) {
    Date SD=
    existingSurvey[i].getDate("DateOfSurvey");
if
(SD.compareTo(dd)>0 && SD.compareTo(Dashd)<0)
{LevelOfNeed=
existingSurvey[i].getInteger("Population");
dd=SD;
}}
return LevelOfNeed;
}}
```

The complete protocol model can be found in [19].

5.6 Validating the KPI properties by using the executable protocol model, goal and conceptual models

The combination of goal, conceptual and executable protocol models in our method enables validation of KPI properties. After the KPIs are modeled as valid numeric algorithms, they can be analyzed and tested. The protocol model can be executed in the Modelscope tool [13].

Below we go through the list of desired KPI properties described in section 2 and illustrate the use of goal, conceptual and protocol models for validation of properties.

1. *A KPI should be in a quantifiable form.*

Quantification means deriving a number or a numerical conclusion from a set of instances of selected concepts in the models.

Protocol modelling has predefined select functions allowing construction of such conclusions.

Function
`selectInState ("BehaviourName","State")`
returns an array of instances, all of which include the specified behaviour.

Function
`selectByRef("BehaviourName","AttributeName")`
returns an array of instances, all of which include the specified behaviour (or object) and have the specified attribute.

The select functions enable modelling of quantifiable KPIs. For example, the `AccessDashboard` applies the select function `selectInState("Survey", "@any")` to gather the set of surveys and further choose the latest survey from this set. From the latest survey the value of the `Population` is read. This value is assigned to the attribute `LevelofNeed` of the `AccessDashboard`.

2. *A KPI needs to be sensitive to changes of the business process state.*

The inputs of the algorithms calculating KPIs are the elements of the state of the underlying business process. If a KPI has not been correctly understood from its definition, the algorithm may mistakenly select wrong instances and attributes.

For example, the source of change of the `KPI1:LevelOfNeed` is the selected instance of the protocol machine `Survey`. The instance is selected using the value of its attribute `DateOf-Survey`. The value of attribute `Population` is assigned to the `KPI1:LevelOfNeed`.

In order to test sensitivity of the `KPI1` to changes of surveys, the protocol model is populated with several instances of `Survey` with different values of `DateOfSurvey` and `Population`. Event `CreateDashboardAccess` is submitted and the `KPI1:LevelOfNeed` is observed. A mistake in the choice of the `Survey` instance will result in the reading of the wrong value of `Population` and the wrong value of `KPI1:LevelOf-Need`.

3. *A KPI should be linear.*

Linearity of a KPI means that the changes of relevant model data and KPI values have a linear relationship.

For example, the
`HI1:  Access Rate = (NumberOfEnteredTreatment/`
`LevelOfNeed)`
represents a linear relation between `HI` and the numerator `NumberOfEnteredTreatment`.

In order to test this, we populate the model with N instances in the state `entered` and submit event `CreateDash-boardAccess`.

Then we populate the model with $N+1$ $(N+2)$ etc. instances in the state `entered` and submit event `Create-DashboardAccess`.

The test results are collected during the execution and the linearity of the relation is analysed.

4. *A KPI should be semantically reliable.*

Execution and demonstration of the model to the users may question semantic reliability of KPIs because the users may have more knowledge than the KPI definition document. Semantic reliability may interpreted as precise definition or meaningfulness.

For example, in our case, the procedure of assessment of the patient's conditions as `Caseness` is not specified in the IAPT document. We present a `Caseness` as 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:
`Recovery Rate = NumberofPeopleMovingToRecovery/`
`(NumberofPeopleCompletedTreatment -`
`NumberOfCasenessPeopleBeforeTreatment).`
We conclude that this KPI is not semantically reliable.

Thus, the demonstration of the model to the users is able to clarify the procedure of the `Caseness` assessment of the patient's conditions.

5. *A KPI should be efficient.*

The KPIs are considered efficient if they are simple, well understood and do not duplicate each other. As we analysed the working programme, the duplications had been already avoided. In IAPT, the `KPI2` and `KPI6a` duplicate other KPIs. They had been found superfluous already by organizations trying to apply the set of IAPT KPIs.

In order to indicate duplication of KPIs, during the modelling the relations between KPIs should be analysed. We have not investigated the use of KPI relations for testing efficiency and consider this as future work.

6. *A KPI should be oriented to improvement, not to conformance to plans.*

The most important property of KPIs is improvement orientation. 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. The value of an improvement oriented KPI cannot be manipulated in the attempt to meet its planned value.

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

The KPI
`HI1:Access Rate = (NumberOfEnteredTreatment/`
`LevelOfNeed)`
is an example of an improvement-oriented KPI. It corresponds to the goal: *"A Referred Person has access to psychological therapies."* The improvement means positive growth of the ratio of treated people to the people needed treatment.

Modelling shows that the numerator and 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 `Number-OfEnteredTreatment` is a summation of individually `Referred Persons`. The numbers of data instances of separated processes grow independently through the model execution. The manipulation of the numerator and denominator of the KPI is unlikely.

Therefore, we conclude that the KPI `HI1:Access Rate` is oriented to improvement.

The KPI
`HI2:Recovery Rate = NumberofPeopleMovingToRecovery/`
`(NumberofPeopleCompletedTreatment -`
`NumberOfCasenessPeopleBeforeTreatment)`
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"*. The improvement corresponds to the growth of the `Recovery Rate`, but the growth may be manipulated by the procedure of the `Caseness` assessment both before and after treatment. If the procedures of the `Caseness` assessment and treatment are assigned to the persons in the same organization, that has an 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.

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 assignment of organizations responsible for the treatment and assessment.

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

6. CONCLUSION

We have presented a method for modelling of KPIs and validation of their properties. The input of our method is a KPI definition document. From this document, we derive a set of related goal, conceptual and protocol models. In comparison with other known methods for KPI modelling [20; 18], our method uses protocol modelling and enables validation of KPI properties already at the stage of KPI definition before the modelling of real business processes of organizations. The validation of KPI properties results in requirements and constrains on business processes.

We have shown that a business behaviour modelling technique is needed for validation of KPI properties. The KPI description documents contain scarce information about fragments of different business processes used for KPI definitions. The behaviour modelling techniques should enable easy synchronization of those fragments, weaving the KPI measurement aspects to the fragments and model execution. Protocol Modelling combines all these properties.

The coexistence of the goal model and the executable protocol model in our method supports reasoning and validation of the properties of KPIs, including semantic reliability and improvement orientation.

In this paper, we have applied our method to a KPI definition document used in the Program for Improving Access to Psychological Therapies (IAPT) [7].

By validation of the property *Semantic reliability* of the KPI `Recovery Rate` we have found that the document does not contain rules for *Caseness* assessment. Applying the KPI, organizations may give different interpretation to undefined rules, and this may result in incomparable KPIs.

By validation of the property *Improvement orientation* of the KPI `Recovery Rate` we have found the need of additional constraints for the business processes.

The case study shows that modelling of KPIs and validation of their properties, before modeling the real business processes, clarify system requirements.

In the future, we plan to investigate the weaving of the abstract process used for KPI definitions into the business processes. Moreover, we plan to further adapt our method for design and analysis of tactical, strategic and complex KPIs [9] used in industry.

Acknowledgement The authors thank the anonymous reviewers of this paper for very useful suggestions for its improvement.

References

[1] R. Allen and D. Garlan. Beyond Definition/ Use: Architectural Interconnection. *Proceedings, Workshop on Interface Definition Languages, Portland Oregon*, 1994.

[2] P. Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof.* Berlin: Kluwer, 2002.

[3] A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. *Sci. Comput. Program.*, 20(1-2):3–50, 1993.

[4] P. Fisk. *People Planet Profit: How to Embrace Sustainability for Innovation and Business Growth.* Kogan Page Limited, UK and USA, 2010.

[5] F.John Reh. Key Performance Indicators (KPI). ABOUT.COM MANAGEMENT. *http://management.-about.com/cs/generalmanagement/a/keyperfindic.htm*, 2012.

[6] C. Hoare. *Communicating Sequential Processes.* Prentice-Hall International, 1985.

[7] Improving Access to Psychological Therapies. IAPT Key Performance Indicator (KPI). Technical Guidance for Adult IAPT Services. *http://www.iapt.nhs.uk/silo/files/iapt-kpi-technical-guidance-201213-v20-.pdf*, 2013.

[8] K. Jensen. *Coloured Petri Nets.* Springer, 1997.

[9] R. Kaplan and D. Norton. Transforming the Balanced Scorecard from Performance Measurement to strategic management: Part I. *Accounting Horizons*, pages 87–104, 2001.

[10] KPIStandard. *http://kpistandard.com/html/EN/Sectors*, 2013.

[11] P. Kueng. Process performance measurement system - a tool to support process-based organizations. *TOTAL QUALITY MANAGEMENT*, 11(1):67–85, 2000.

[12] A. McNeile and E. Roubtsova. CSP parallel composition of aspect models. In *Proceedings of the 2008 AOSD workshop on Aspect-oriented modeling*, AOM '08, pages 13–18, New York, NY, USA, 2008. ACM.

[13] A. McNeile and N. Simons. http://www.metamaxim.com/. 2005.

[14] A. McNeile and N. Simons. Protocol Modelling. A Modelling Approach that Supports Reusable Behavioural Abstractions. *Software and System Modeling*, 5(1):91–107, 2006.

[15] S. Mount, M. Hammoudeh, S. Wilson, and R. Newman. CSP as a domain-specific language embedded in Python and Jython. *Welch et al.[248]*, pages 293–309, 2009.

[16] OMG. *Unified Modeling Language: Superstructure version 2.1.1 formal/2007-02-03.* 2003.

[17] C. Petri and W. Reisig. *Petri net.* Scholarpedia 3(4):6477., Retrieved 2008-07-13.

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

[19] E. Roubtsova. Protocol Model of the KPIs from "The programme "Improving Access to Psychological Therapies. *http://www.open.ou.nl/elr/IAPT.zip*, 2013.

[20] S. Strecker, U. Frank, D. Heise, and H. Kattenstroth. 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.