Summary

A ‘Key Performance Indicator’ (KPI) is defined as “a metric used to quantify the efficiency and/or effectiveness of an action that is compared to a given norm or target which is derived from business strategy”. KPIs should be specified, modeled and validated to have specific properties. These required properties are: quantifiability, sensitivity, congruency, reliability, efficiency and improvement-orientation. But with standard and popular modeling notations, such as those in ‘Unified Modeling Language’ (UML), the properties cannot be guaranteed.

In this work, recommendations on properties are combined with a constructive definition into a conceptual model of KPIs and their required properties. This model can be used for the specification, modeling and validation of KPIs. Also, the modeling semantics that are needed to support the modeling and validation of KPIs are explained. The modeling notation that is used should not only include semantics of business objects and their attributes, but also of states and events that are extended with data to make up transitions in the object-lifecycles, and CSP-composition of the transitions to compose executable and deterministic behavior. The semantics are included in the method EXTREME which uses ideas from approaches to goal modeling and Protocol Modeling.

As a case study, the time-related KPI ‘Percentage of In-Time Processed Orders’ of business process ‘Order to Pay’ is specified, modeled and validated using the conceptual model and a protocol model. The KPI is initially specified as the percentage of closed orders for which the ‘Processing Lead Time’ equals or is lower than the ‘Norm Lead Time’. Using the conceptual model this is further specified. By developing and executing a protocol model, it is validated that the KPI has its value calculated from the right number of instances of business concepts (quantifiability); changes in states and attributes-values of the instances (sensitivity); a clear mathematical relationship (congruency); and a minimum number of assumptions (reliability) and scenarios for manipulation (improvement-orientation).

Future research is needed on the risk that KPIs in general or in specific (time-related) cases might be used to manipulate and/or do not reflect real performance or improvement.

Key words: requirements, properties, characteristics; performance measures, performance indicators, key performance indicators (KPIs); and/or events, states and object-lifecycles.
Contents

1. Research Problem ........................................................................................................................................... 5
2. Questions and Approach .................................................................................................................................. 5
3. Literature Study: KPIs and their Required Properties .................................................................................... 7
   3.1. Definition of KPIs ...................................................................................................................................... 7
   3.2. Required Properties of KPIs ................................................................................................................... 7
      3.2.1. Required Properties of Measures or Metrics in General ................................................................. 7
      3.2.2. Recommended Properties of Performance Measures ................................................................. 8
      3.2.3. Required Properties of KPIs ........................................................................................................... 9
   3.3. A Conceptual Model of KPIs and their Properties .................................................................................... 11
      3.3.1. Building the Conceptual Model ........................................................................................................ 11
      3.3.2. Categorization of Properties ........................................................................................................ 11
4. Literature Study: Modeling and Validation of KPIs ......................................................................................... 12
   4.1. Modeling Semantics .................................................................................................................................... 13
   4.2. Standard and Popular Modeling Notations .............................................................................................. 14
   4.3. EXTREME and Protocol Modeling ........................................................................................................ 15
5. Case Study: KPI ‘Percentage of In-Time Processed Orders’ ........................................................................... 16
   5.1. Goals and Requirements ....................................................................................................................... 17
   5.2. Specification .............................................................................................................................................. 18
   5.3. Protocol Model ....................................................................................................................................... 19
      5.3.1. Sending and Confirming Orders ........................................................................................................ 20
      5.3.2. Registering and Changing Invoices ................................................................................................. 20
      5.3.3. Paying Invoices and Closing Orders ............................................................................................... 22
      5.3.4. Creating Dashboards ....................................................................................................................... 23
   5.4. Validation ................................................................................................................................................... 24
6. Conclusion and Future Work .......................................................................................................................... 25

References .............................................................................................................................................................. 28
Appendix A: Relating Properties and Concepts .................................................................................................. 31
Appendix B: Code of the Protocol Model .......................................................................................................... 34
   Protocol Model .............................................................................................................................................. 34
   Callback Functions ....................................................................................................................................... 36
Appendix C: Test-Cases used during Validation ................................................................................................. 40
1. Research Problem

According to the literature on performance management, Key Performance Indicators (KPIs) are required to have specific properties. However, with the modeling semantics that are used in the standard and popular modeling notations of the current practice, there is no guarantee that KPIs will have the required properties; based on inadequate modeling semantics, the KPIs in the current practice run the risk of being unreliable or plan-oriented. It is claimed that the method of the ‘Executable Requirements Engineering Management and Evolution’ (EXTREME; Roubtsova, 2013) offers a solution by combining ideas from goal modeling and Protocol Modeling, i.e. by using suitable modeling semantics, thereby enabling complete and correct specification, modeling and validation of KPIs (Roubtsova & Michell, 2013a, 2013b, 2014). More experimental evidence still is needed on the use of EXTREME in the context of KPIs and their required properties.

2. Questions and Approach

The current research aims to provide additional experimental evidence on how EXTREME is used for the complete and correct specification, modeling and validation of KPIs, namely by replicating earlier research (Roubtsova & Michell, 2013a, 2013b, 2014), but with a new case study and while gaining insights into the properties of KPIs and the modeling semantics. The new case study involves a time-related process KPI ‘Percentage of In-Time Processed Orders’ of business process ‘Order to Pay’, which is not considered in earlier research. The main question of the current study is formulated using CIMO-logic (Denyer, Tranfield & van Aken, 2008; van Aken, Chandrasekaran & Halman, 2016):¹

*Does application of the method EXTREME [I], with its use of modeling semantics [M], enable the complete and correct specification, modeling and validation [O] of time-related KPIs [C]?*

In the context of this research, “complete and correct” should be interpreted as KPIs having all of the required properties and thus not running the risk of being for instance “unreliable” or “plan-oriented”. Specifically this means that there should be no scenarios of for instance non-reliability - or if there are, that such scenarios are found using the method EXTREME.

The sub-questions of this research are the following:

1. What are KPIs and their required properties?
2. What are suitable semantics for the specification, modeling and validation of KPIs?
3. What are the required properties of KPI ‘Percentage of In-Time Processed Orders’?
4. How is EXTREME applied for the KPI ‘Percentage of In-Time Processed Orders’?

The above main and sub-questions are answered by way of these following methods:

A. Literature study (sub-questions 1 and 2), including the ordering of domain concepts and the classification of the properties of KPIs within a conceptual model.
B. Design experiment (sub-questions 3 and 4): specification, modeling and validation of the KPI ‘Percentage of In-Time Processed Orders’.
C. Analysis and discussion of the results of A and B.

The original intent of the research was to first carry out a literature study to answer the sub-

---

¹ The CIMO-logic might be formulated by using the following sentence-template: “for this problem-in-Context [C] it is useful to use this Intervention [I], which will produce through these Mechanisms [M] this Outcome [O]” (Denyer, Tranfield & van Aken, 2008; van Aken, Chandrasekaran & Halman, 2016).
questions 1 and 2, and secondly to have a design experiment to answer sub-questions 3 and 4. The answering of sub-question 1 proved to be too difficult, however, because of a lack of categorization of properties of KPIs found in literature. Therefore, in order to be able to fully answer sub-question 1, it was decided to first build a conceptual model of properties of KPIs found in literature, before proceeding to the second part of the literature study and the design experiment thereafter – that is: before answering the sub-questions 2, 3 and 4. Accordingly, the final approach to the collection of data is twofold in the current research.

First, a literature study is carried out in the field of performance management in order to answer sub-question 1 and to build a conceptual model of (required) properties of KPIs that are found. Literature is searched for theory on KPIs and their required properties (Figure 1, top left) using the terms “requirements”, “characteristics”, “properties” and/or “performance measures”, “performance indicators” and “key performance indicators”. Domain concepts resulting from the search are then used to build a conceptual model, which in turn is used for categorization of the properties of KPIs found in literature (see also Appendix A). This first part of the research concerns the collecting of data on KPIs and their properties in general.

Secondly, again a literature study is carried out this time in the fields of business process management and behavioral modeling to answer sub-question 2. The literature is searched for theory on modeling and validation for which the standard and popular modeling notations in the current practice do not have adequate semantics (Figure 1, bottom left), using terms such as “events”, “states” and “object-lifecycles”. The results of the search are used together with the results of the first part of the research to formulate a hypothesis on how the method EXTREME enables the complete and correct specification, modeling and validation of KPIs.

The hypothesis is tested in a case study with a design experiment for the time-related KPI ‘Percentage of In-Time Processed Orders’ (Figure 1, middle) of business process ‘Order to Pay’ - which is considered to be representative of the business processes that use time-related KPIs. In the design experiment this KPI and its properties are specified, modeled and validated using EXTREME (see Appendix B for code of the protocol model) and in doing so, sub-questions 3 and 4 are answered. This second part is thus aimed at the collecting of data on the KPI ‘Percentage of In-Time Processed Orders’ and its required properties in specific.

The case study is to show how KPIs should be specified, modeled and validated to have the required properties by applying the method EXTREME; it is to demonstrate how, with a use of EXTREME, gaps can be found in the initial specifications and/or process model of a KPI with regard to required properties - e.g. how scenarios of non-reliability or plan-orientation can be found. As this research is a replication study, the setup of the case study is similar to that in the earlier research (Roubtsova & Michell, 2013a, 2013b, 2014): the validation of the KPI ‘Percentage of In-Time Processed Orders’ and its properties is done by executing a
specific protocol model of the KPI and by using the data of the model runs (see Appendix C).

Results of the literature study and case study are analyzed and discussed to answer the main research question (Figure 1). In the discussion we move beyond the scope of this study to question whether a validated KPI can indeed be considered “complete and correct”.

3. Literature Study: KPIs and their Required Properties

In the following the results of the first literature study are given, namely: a definition of KPIs and an overview of properties of KPIs found in the literature on performance management. This is to give an answer to sub-question 1: What are KPIs and their required properties?

3.1. Definition of KPIs

In literature on performance management, the terms ‘performance measure’, ‘performance indicator’ and ‘key performance indicator’ (KPI) are usually considered to be synonymous, which is confusing. Before giving a definition of the latter, the former should be explained.

A performance measure is a “metric used to quantify the efficiency and/or effectiveness of an action” (Neely, Gregory, & Platts, 1995, 2005). Simply put, the efficiency of an action is its ability to avoid waste, while its effectiveness is its ability to produce a desired result. In this context such an action is part of a business process that leads to performance and that needs to be measured through a “process of quantification”. The measure might for instance relate to the cost or the time of an action, such as a ‘Processing Lead Time’. But in order to get a true indication of the performance, more is needed than such a performance measure alone.

In order to get a performance indicator the measure must be compared to “a given norm or target” (Lohman, Fortuin, & Wouters, 2004). For example, in the business strategy of ‘just-in-time’ processing, the processing of orders just too late (or just too early) is seen as a waste. Hence, a performance measure ‘Processing Lead Time’ should be compared to a norm or target, such as a ‘Norm Lead Time’, in order to get an indication of the waste. Key is that this norm or target for the measure is derived from business strategy (Eckerson, 2009).

Taken together, a key performance indicator (KPI) can be defined as a metric used to quantify the efficiency and/or effectiveness of an action that is compared to a given norm or target which is derived from business strategy. This definition from literature on performance management gives a basic understanding of KPIs and a starting point for their specification, modeling and validation. Following the definition, for instance, a KPI ‘Percentage of In-Time Processed Orders’ of the business process ‘Order to Pay’ may initially be specified as: “the percentage of closed orders for which the ‘Processing Lead Time’ equals or is lower than the ‘Norm Lead Time’.”

More is needed for complete and correct specification, modeling and validation of KPIs than this basic definition and such an initial specification alone, however; KPIs should be further specified, modeled and validated to have some required properties - which should hence already be taken into account, before the implementation of a KPI. This is explained next.

3.2. Required Properties of KPIs

3.2.1. Required Properties of Measures or Metrics in General

The two most important properties of measures or metrics in general are their reliability and validity (Gregory, 2004; Rosnow & Rosenthal, 2002; Saunders, Lewis & Thornhill, 2015). Reliability means the degree of consistency or stability of a measure, i.e. whether a
measure can be repeated and confirmed by further measures. Validity refers to whether a measure indeed measures what it is supposed to measure. This means that evidence is needed with regard to: content-related validity (i.e. that a measure represents and covers all aspects of the material that it is supposed to represent); criterion-related validity (i.e. that the measure correlates with outcome criteria); and construct validity (i.e. the ability of a measure to discriminate, to not correlate with other measures of some related but different material).\(^2\)

Note that the reliability of a measure is a necessary but insufficient precursor of its validity.

Both of the above properties are interdependent on the property of objectivity, which refers to ruling out of measurement errors due to (inter)subjectivity - and which is achieved by ways of “measurement and quantification” (Reiss & Sprenger, 2014). An important factor of influence on the reliability and validity of a measure, moreover, is the observer effect: the effect that the process of measuring alone changes the material that is being measured. In this context this means that if people participating in a business process become aware that their behavior is measured, they change their behavior (Saunders, Lewis & Thornhill, 2015). This effect is addressed in the discussion of this research with regard to “Goodhart’s Law”.

The above properties are mentioned in literature on the performance management along with a fairly huge amount of other recommended or required properties of performance measures and/or KPIs - they are not mentioned as the single most important ones, however. That is, although there is consensus in the literature that KPIs are required to have specific properties, there is no agreement on exactly which ones and on how the different properties actually relate. Different authors provide different properties without a clear categorization.

### 3.2.2. Recommended Properties of Performance Measures

One early study of performance measures (Neely, Richards, Mills, Platts, & Bourne, 1997) provides a review of a list of recommendations with regard to their design, that are made in the early literature on performance management. Some of the recommendations are said to be referring to the “characteristics of well-designed performance measures”. For instance, performance measures should be “derived from strategy”, “related to specific goals (targets)” and “based on quantities that can be controlled” - which is in line with the above definition of KPIs. Furthermore, for instance, the performance measures should be simple to understand, provide timely and accurate feedback, focused on improvement, and based on an explicitly defined formula and source of data.

In later literature on performance management (Alwaer & Clements-Croome, 2010; Bassen & Kovacs, 2008; Crampton et al., 2004; Garengo, Biazzo, & Bititci, 2005; Kueng, 2000; Kumar et al., 2013; Mian, Humphreys, & Sidwell, 2004; Sheldon, Abercrombie, & Mili, 2009; Takim & Akintoye, 2002; Veleva & Ellenbecker, 2001) most early recommendations on characteristics of performance measures are corroborated, one way or another, while even more are provided. Additional recommendations are for instance that performance measures should be “easily measurable”, “inexpensive in time and cost” and also “sensitive to change”.

For this study a ‘longlist’ of recommended properties of performance measures is extracted from the literature on performance measurement. That is, performance measures should be:

1. Derived from strategy.
2. Simple to understand.
3. Timely and accurate feedback.
4. Based on quantities that can be controlled.
5. Reflect the business process.

\(^2\) Additional types of validity in the context of experimental research are, for instance, internal validity (ruling out of plausible rival hypotheses), and external validity (generalizability of the research results).
6. Related to specific goals (targets).
7. Relevant.
8. Part of a closed management loop.
9. Clearly defined.
11. Focus on improvement.
12. Maintain significance, consistent in time.
14. Have explicit purpose.
15. Based on explicitly defined formula and source of data.
16. Employ ratios instead of absolute numbers.
17. Use data which are automatically extracted as part of a process.
18. Reported in simple, consistent format.
20. Provide information.
21. Precise (exact about what being measured).
22. Objective (not based on opinion).
23. Easily measurable.
24. Inexpensive in time and cost.
25. Sensitive to change.

Still, however, no agreed-upon list of required properties of performance measures or KPIs is found in the literature on performance management; there is no clear distinction between different recommendations and no clear categorization of properties on the above longlist.

3.2.3. Required Properties of KPIs

Because of the lack of clear categorization of the above longlist of properties, it is still too early to be able to fully answer sub-question 1: What are KPIs and their required properties? Therefore we build a conceptual model of properties of KPIs on the basis of a ‘shortlist’ of properties. The aim is to provide a means for categorizing the properties found in literature.

Current research often uses the shortlist of required properties of KPIs (Kitchenham, 1996; Kueng, 2000; Winchell, 1996) which is also used in the research on Protocol Modeling and KPIs (Roubytsova & Michell, 2013a, 2013b, 2014) - and which seems to capture the most important required properties of KPIs. The required properties of KPIs on this shortlist are: quantifiability, sensitivity, congruency, reliability, efficiency and improvement-orientation.

Before proceeding, these required properties of KPIs are explained from a perspective of (behavioral) modeling, as KPIs should be validated on the basis of some process model.

Quantifiability. KPIs should be measured through a process of quantification. The process of quantification refers to an algorithm that is based on first-order (predicate) logic for selecting and counting the number of relevant instances in a model. These are instances of business concepts that are included in a model to support a business process and which might be relevant to a KPI because they are in particular states or have particular attribute-values. The count of relevant instances should be used for calculation of a KPI. If a KPI does not allow for such an algorithm, it is not considered quantifiable and should be transformed.

Sensitivity. KPIs should be sensitive to changes in states and attribute-values. The value of a KPI should change only as a result of changes in the states and attribute-values of relevant instances of business concepts. Sensitivity is expressed by the minimum magnitude of change in the states or attribute-values which is required to produce a change in the value of the KPI. However, although a KPI should have some sensitivity to make sure that the changes in the states or attribute-values are detected, high sensitivity is not always required, as KPIs often only need to identify trends or problems in performance (Kumar et al., 2013).
Congruency. KPIs should be congruent with changes in states and attribute-values. Changes in the value of a KPI should be proportional to the changes in the states and the attribute-values of the relevant instances: small and big changes in the states and attribute-values should lead to small and big changes in the value of the KPI, respectively. Moreover, there should be a clearly defined mathematical relationship between the value of a KPI and the states and attribute-values of instances - which must be able to be validated. Different mathematical relationships can be used, but the relationship should be easy to understand.  

Reliability. KPIs should be semantically reliable and free from measurement error. The values of a KPI should be consistent, more or less the same in similar circumstances, and calculated correctly in both routine and unexpected circumstances. A reliable KPI is free from errors of measurement. If, for instance, a KPI is measured by several different experts, then subjective factors should not be a source of error. Reliability can be measured by the number of additional, possibly incorrect or incomplete assumptions about the business process or the KPI formula that are needed to derive a KPI value. A KPI is considered to be reliable if there are no additional assumptions.

Efficiency. KPIs should not be a waste of effort from a cost/benefit point of view. As measurement, calculation and understanding of KPIs require human, financial and physical resources, a KPI should be worth the effort. In order to avoid a waste of effort, a KPI should be inexpensive in time and cost, easily quantifiable, created in the simplest way from any constituent measures, calculated in the simplest possible way, and easily understood. In the context of the current work, this property is considered to be mostly subjective and in need of a different approach. Therefore, although this property is used in the following for building a conceptual model, the property is not used for further specification, modeling and validation.

Improvement-Oriented KPIs. KPIs should focus on improvement, not conformance. KPIs should be oriented to improvement rather than conformity with plans. They should be related to business strategy and goals, while creating an atmosphere in which feedback is viewed in a positive light - this is not achieved though measures of violations or errors. There is a risk that a KPI is used to manipulate (Fisher & Downes, 2008; Meyer, 2005), which often is the result of incomplete assumptions about the business process. Orientation to improvement is measured by the number of manipulative scenarios for a KPI. A KPI is considered to be oriented to improvement, if there are no such scenarios in the underlying business process.

It should be noted that the shortlisted properties are related to the already mentioned properties of reliability and validity and, in a broader sense, to that of objectivity. While this is clear for the shortlisted property of reliability, it is true for other properties. The properties of quantifiability, sensitivity and congruency relate to the property of validity, as these three are content- en criterion-related. And most notably the properties of reliability and improvement-orientation hold a risk that performance is not reflected objectively, as a result of possibly incorrect or incomplete (subjective) assumptions about the business process or KPI formula. This risk should be taken into account during specification, modeling and validation of KPIs. Now that the shortlisted properties are explained, it is still not clear how they can be used for the categorization of the (longlisted) properties found in the literature or how these properties can and should be used during the further specification, modeling and validation of KPIs.

3 Originally this requirement referred to “linearity” of KPIs (Kitchenham, 1996; Kueng, 2000; Winchell, 1996). Later the requirement was alternatively stated as follows: “the value of a KPI must be able to be shown by a consistent mathematical relationship in its simplest form” (Roubtsova & Michell, 2013b). Here this is adapted to “congruency” according to the recommendation that KPIs should be relative and employ ratios in order to be easily understood (Garengo et al., 2005; Neely et al., 1997).
3.3. A Conceptual Model of KPIs and their Properties

In the following first it is briefly explained how the conceptual model is built and second how it is used for a categorization of properties of KPIs found in literature (see also Appendix A).

3.3.1. Building the Conceptual Model

The conceptual model is built by addressing the shortlisted properties with the concepts that are used in a constructive definition of KPIs that is provided by Roubtsova & Michell (2013a, 2013b, 2014) and which is in countable and comparable terms (Figure 2, the concepts are depicted with white rectangular boxes). Some additional concepts are used to extend the conceptual model to get a full representation of the properties (Figure 2, these additional concepts are depicted with gray rectangular boxes). The building of the conceptual model concerns the preparation of the collecting of data from literature on KPIs and their properties.

The following constructive definition is provided (in literature on behavioral modeling): “A KPI is a cumulative function of the cardinality of a set of selected business objects [from a business process] and the values of their attributes. The selected business objects give the value true to the selection predicate. The selection predicate compares the state and attribute values of each business object with the border values of attributes, the moment of selection and the time interval” (Roubtsova & Michell, 2013a, 2013b, 2014).

Now for instance the property of quantifiability is addressed with the “cardinality of a set of selected business objects and the values of their attributes” (Figure 2, left). This latter is the number of objects of this set. And improvement-orientation for instance demands the extension of the model with the concepts of ‘KPI Value’, ‘Improvement’, ‘Goal’ and ‘Strategy’ (Figure 2, middle). These show that improvement is seen as the result of comparing the KPI values taken from at least two different time periods. The conceptual model now shows how all of the shortlisted properties are addressed with these (additional) concepts (Figure 2).

3.3.2. Categorization of Properties

The conceptual model is used for the categorization of the properties found in the literature by relating the longlisted properties to concepts of the model – this is the actual collecting and analyzing of data found in literature on KPIs and their properties in general. Relating the properties to concepts forms the proof of correctness of the shortlist of required properties.

The longlisted properties are classified into three groups of shortlisted properties (Figure 2, each group is depicted with a dashed round-cornered box) and each longlisted property is placed in one of the groups if the property uses concepts of the depicted part of the model or if it forms a tautology with the concepts of the depicted part (that is: to say the same thing twice in different words). Most of the longlisted properties for instance belong to the first group of “Quantifiability+Sensitivity+Congruency+Reliability” as these are related to the concepts of ‘Process’ and of ‘KPI formula’ (Figure 2). For example, property #4 uses the term “quantities” which refers to the number of selected business objects and/or attribute-values, which are taken directly from the business process and hence can be “controlled”. Properties #1, #6, #7, #11 and #14 are in the second group of “Improvement-Oriention” and it is clear that these relate ‘KPI Value’ to ‘Improvement’, ‘Goal’ and ‘Strategy’ (Figure 2).

An explanation of all the relations between properties and concepts is given in Appendix A.
Figure 2 Conceptual model of properties found in literature on performance management.

The above conceptual model shows that the required properties of KPIs are quantifiability, sensitivity, congruency, reliability, efficiency and improvement-orientation, as these can be used for the categorization of the properties of KPIs found in literature. Moreover, it shows that, among other concepts, the ‘Business Objects and […] Attributes’ and of ‘Border Values of Attributes’ should be involved in the specification, modeling and validation of KPIs. Indeed this is the case for the KPI ‘Percentage of In-Time Processed Orders’; its initial specification already gives the ‘Processing Lead Time’ and ‘Norm Lead Time’ as time-related instances of the respective concepts - which also shows how the KPI is classified as a time-related one.

This is not to say, however, that the above model together with an initial specification of a KPI is sufficient for the complete and correct specification, modeling and validation of a KPI. The model shows that concepts of for instance business objects and their attributes are involved – and hence that the modeling notation that is used for the modeling and validation of KPIs should include some corresponding semantics. But additional semantics are needed, to support a process model of a KPI that is executable and extended with data, in order to be able to calculate actual KPI-values and enable the validation of the properties of the KPI - by running a model and obtaining values, incorrect or incomplete assumptions about a business process or KPI formula can be revealed.

4. Literature Study: Modeling and Validation of KPIs

In the following the results of the second literature study are given, that is: an overview of the semantics that are needed for the specification, modeling and validation of KPIs, according to literature on business process management and behavioral modeling. This is to answer sub-question 2: What are suitable semantics for the specification, modeling and validation of KPIs?
4.1. Modeling Semantics

As said, the modeling notation that is used for the modeling and validation of KPIs should support executable models that are extended with data in order to be able to calculate actual KPI-values, and hence to be able to validate the properties of the KPI. Moreover, the models should be deterministic, in the sense that the executions should be repeatable: if a model is executed twice from some initial state with the same processing, then the final state should twice be the same. This latter is not necessarily true with the standard and popular modeling notations. It demands that a modeling notation includes semantics not only of the business objects and their attributes, but also of “states”, “events” to trigger transitions, and a proper “composition of behaviors” that is constructed from these transitions (McNeile & Roubtsova, 2008a, 2010; Roubtsova, 2015).

A state is a situation in which the instances of business objects can exist for “some period of time”; an event is a significant happening of “minimal duration” that triggers them to move into a state (Dori, 2011). States and events make up transitions in the object-lifecycles; instances move from one state to another state as explicitly triggered by the events (Cohn & Hull, 2009; Lohmann, 2011; Pedrinaci, Domingue, & de Medeiros, 2008; Roubtsova, 2015). In the business process ‘Order to Pay’, for instance, an order is needed to be able to obtain the “process algebras” of CCS (Milner, 1980) and of the ‘Calculus of Communicating Sequential Processes’ or CSP (Hoare, 1985).

The CCS-composition allows for concurrent execution of transitions in an interleaving manner, that is: one after another. In this case, the business objects are represented by machines called ‘transducers’ which can both send and receive events in order to initiate the transitions: one machine first sends an event and initiates a transition, and another machine then receives this event and initiates another transition. A consequence of CCS-composition is that events might be stored in buffers and compete for handling by receiving machines, as it is determined by outside factors which events are received first (such as the time and path of the signal-transfer). This contributes to the non-determinism of the constructed behaviors.

The CSP-composition allows for concurrent execution of the transitions in a parallel manner, that is: synchronously. In this case, there are machines called ‘receivers’ which can only receive events within an environment that sends events: the environment first sends an event and several machines then receive the event and initiate multiple transitions, but only if all machines that recognize the event are ready to accept it – otherwise, it is refused and no transitions occur. A consequence of the CSP-composition is a so-called “observational consistency”: properties of the behavior as a whole can be deduced from knowledge of the behaviors of local machines. Moreover, the behavior of a CSP-composition is deterministic.

Both states and events should be extended with data in order to be able to obtain the values of KPIs. In ‘Labeled Transition Systems’ (LTS), for instance, events are extended to be messages of a given structure for communication of data between machines, and states are extended to have data-attributes with which the messages are related to the instances of machines (Roubtsova, 2015). The data is needed to be able to obtain values of performance measures and norms or targets of the efficiency and/or effectiveness of actions (which are signified by events), to compare the values of these and to calculate the values of the KPIs.

Thus, the modeling and validation of KPIs demands a modeling notation that not only has semantics of business objects and their attributes, but also of states and events in order to make up transitions in the object-lifecycles and of CSP-composition for the construction of deterministic behavior out of these transitions. If the latter is not present, then the behavior of
a model can only be validated statistically. Furthermore, these states and events should be extended with data in order to be able to obtain values of KPIs. The standard and popular modeling notations of the current practice, however, do not include the suitable semantics.

4.2. Standard and Popular Modeling Notations


State machine diagrams exist in two variants in UML. The ‘Behavioral State Machine’ (BSM) or “statechart” uses states and events to make up transitions in object-lifecycles, but also CCS-composition due to which resulting behaviors are non-deterministic and validation must be done statistically. The ‘Protocol State Machine’ (PSM) also uses states and events to make up transitions in object-lifecycles, but for unexpected events the resulting behaviors are undefined and validation must also be done statistically (Roubtsova, 2015). Therefore, these variants of state machine diagrams in UML are not suitable for the validation of KPIs.

Activity diagrams in UML are used in the context of performance management (Sánchez González et al., 2010), although there is a need to extend them with goals and performance measures (Korherr & List, 2006). The activity diagrams in UML use transitions, but these show the flow of the activities in a business process or “workflow”; states are mostly hidden in the activities and the transitions do not require explicit triggering by events (Roubtsova, 2015). Because it is difficult to relate KPIs to the activities and transitions, it is difficult to validate the KPIs and their required properties on the basis of the activity diagrams in UML.

Petri Net, EPC and BPMN are similar to the activity diagrams in UML, as states in BPMN are mostly hidden in activities, and the transitions in Petri Net, EPC and BPMN do not require explicit triggering by events. The modeling and validation of KPIs is done in an extension of Petri Net called ‘Colored Petri Net’ (CPN) and KPIs can be related to countable ‘tokens’ in CPN. But these tokens are taken from random states (called “places”), when the transitions are triggered, due to which the resulting behavior of CPN is non-deterministic and validation of KPIs and required properties can only be done statistically (Roubtsova, 2015).

Class Diagrams in UML, finally, are the most used and informative of all of the above notations, but client involvement is not high (Dobing & Parsons, 2006) and they are not used for performance management (Sánchez González et al., 2010). They are static models with objects that show the structure of a system, that do not have states and events as elements. States and events can be modeled in class diagrams, but then the same behavior will be represented multiple times in different objects, possibly with inconsistent views (McNeile & Simons, 2006). Hence, they are also not suitable for the modeling and validation of KPIs.

As an illustration the business process of ‘Order to Pay’ is modeled as an activity diagram in UML and a diagram in EPC (without KPI ‘Percentage of In-Time Processed Orders’, Figure 3).

---

4 For instance, each event would have its own class with the responsibility for checking if the objects and states required for the events exist. If the same checks are needed for different events, then they need to be modeled multiple times, but there is no guarantee that they are modeled in the same way.
Figure 3 Examples of an activity diagram in UML (left) and an EPC (right) of ‘Order to Pay’.

The activity diagram shows the transitions between activities such as the sending and the confirming of orders (Figure 3, left). There are no events, and states such as orders being ‘sent’ and ‘confirmed’ are hidden (except for initial and end-states represented by circles). The EPC shows the transitions between these activities, but with the inclusion of so-called events such as orders being ‘sent’ and ‘confirmed’ (Figure 3, right). Semantically, these are not events but states, which is a confusion of events and states that is shown by mapping events in EPC onto states/places in Petri Net (Van der Aalst, 1999) - there are no events.

Taken together, the standard and popular modeling notations have limitations with regard to the modeling and validation of KPIs. The latter requires that the process models of KPIs are executable, deterministic and extended with data, but the standard and popular notations do not have the required semantics. They do not have a proper semantic notion of states and events to make up transitions in the lifecycles of the business objects, nor are the states and events extended with data, and their mechanisms for composition result in non-deterministic behaviors of models. There is a “semantic mismatch” (Roubtsova, 2013), as the semantics of the standard and popular modeling notations are not adequate for the validation of KPIs.

4.3. EXTREME and Protocol Modeling

The method EXTREME offers a solution to the semantic mismatch by exploiting similarities in the semantics of approaches to goal modeling and Protocol Modeling, thereby enabling the complete and correct specification, modeling and validation of KPIs. More specifically, EXTREME aims to do this through the identification of incomplete or incorrect assumptions
about for instance the business process or KPI formula, with the data obtained by executing a protocol model (Roubtsova, 2013; Roubtsova & Michell, 2014). The semantics of Protocol Modeling include states and events that are extended with data, and the deterministic CSP-composition (McNeile & Roubtsova, 2008a, 2008b, 2009, 2010; McNeile & Simons, 2006).

The so-called “protocol machines” of Protocol Modeling have states they can be in, repertoires of events that they can recognize, and local storages. The protocol machines are composed using the CSP-composition to model object-lifecycles, resulting in deterministic behavior of the model: when events are presented from the environment, they are accepted if and only if all protocol machines that recognize the same events are able to accept them. Otherwise, events are refused (or ignored). As a result, accepting protocol machines move from one state in object-lifecycles to another and/or update the data in their local storages.

Accordingly, the hypothesis is that method EXTREME has suitable modeling semantics that enable complete and correct specification, modeling and validation of KPIs. This hypothesis is tested in the following case study, with a design experiment for the KPI ‘Percentage of In-Time Processed Orders’ of the business process ‘Order to Pay’ using its initial specification.

First, it should be possible to obtain a ‘Processing Lead Time’ of business process ‘Order to Pay’, for instance by composing protocol machines to model the business concept of ‘Order’ and to recognize events ‘Send’ and ‘Close’. When the event ‘Send’ is presented and accepted, an instance of ‘Order’ should move to state ‘Sent’ and write a ‘Date Sent’ as an attribute to its local storage. When the event ‘Close’ is presented, similarly, the ‘Order’ should move to state ‘Closed’ and write a ‘Date Closed’ to its local storage. Now, from the data in the local storage it should be possible to calculate a ‘Processing Lead Time’.

Second, a business concept ‘Dashboard’ might be modeled to have the attributes ‘Closed Order Count’, ‘Norm Lead Time’ and ‘Percentage of In-Time Processed Orders’. The value of ‘Closed Order Count’ should be set to automatically increase as instances of ‘Order’ move to state ‘Closed’. If the value of ‘Norm Lead Time’ is specified, this can be used to derive a “count of orders that are ‘closed’ and for which the ‘Processing Lead Time’ equals or is lower than the ‘Norm Lead Time’”. Using all attributes in the local storages of protocol machines, it should be possible to calculate values of KPI ‘In-Time Processed Orders’ on a ‘Dashboard’.

By running the protocol model and obtaining the KPI-values, it should be possible to identify incomplete or incorrect assumptions about for instance the business process ‘Order to Pay’ or the KPI formula for the KPI ‘Percentage of In-Time Processed Orders’. If there are any scenarios of for instance non-reliability or plan-orientation, these should be found using EXTREME. That is, it should be possible to validate whether the KPI has all of the required properties of quantifiability, sensitivity, congruency, reliability, efficiency and improvement-orientation – which should be further specified on the basis of the requirements for the KPI.

5. Case Study: KPI ‘Percentage of In-Time Processed Orders’

As preparation for the collecting of data, first the goals and requirements for KPI ‘Percentage of In-Time Processed Orders’ of the business process ‘Order to Pay’ are given, followed by a further specification of the KPI and its required properties. This is to answer sub-question 3: What are the required properties of KPI ‘Percentage of In-Time Processed Orders’? Next, a protocol model is developed of the KPI and the business process - which is to finalize this preparation, but also to demonstrate the semantics of EXTREME using Protocol Modeling.

Concerning the actual collecting of data, the protocol model is executed to obtain the values of the KPI ‘Percentage of In-Time Processed Order’ and to validate whether there are no incomplete or incorrect assumptions about for instance the business process or the KPI formula. This is to answer sub-question 4: How is EXTREME applied for the KPI ‘Percentage
of In-Time Processed Orders’? In all, of course, this is to demonstrate how the method of EXTREME enables the complete and correct specification, modeling and validation of KPIs.

5.1. Goals and Requirements

The goals and requirements for KPI ‘Percentage of In-Time Processed Orders’ are shown in Figure 4. There is one goal, namely the “in-time processing” of business process ‘Order to Pay’. Both the business process and its performance management should be supported, including the lifecycles of orders and invoices (Figure 4, top) and of a dashboard presenting the KPI ‘Percentage of In-Time Processed Orders’ (Figure 4, bottom). Requirements are that a logistical employee can send and confirm orders and that a financial employee can register and pay invoices. Specific business rules are:

- Only orders that are sent might be confirmed.
- Only invoices for confirmed orders might be registered. Several invoices might be registered for one order, but their total amount should not exceed the order amount.
- Only invoices that are registered might be paid.
- A business manager should be able to set and change the ‘Norm Lead Time’ in days.

![Figure 4](image)

Figure 4 The goals and requirements for KPI ‘Percentage of In-Time Processed Orders’.

For the sake of simplicity some typical business requirements are excluded, such as those regarding the receipt and further handling of the goods that are ordered. Requirements that
are of a more technical nature, such as those concerning the formats, lengths and units of attributes (integers vs. strings, number of characters, specific currencies) are also excluded.

5.2. Specification

The KPI ‘Percentage of In-Time Processed Orders’ needs some further specification before it can be modeled and validated. Initially the KPI was specified as “the percentage of closed orders for which the ‘Processing Lead Time’ equals or is lower than the ‘Norm Lead Time’”. The conceptual model of KPIs and their required properties is used for further specification.

In order to be able to guarantee quantifiability the ‘Processing Lead Time’ should be further specified. Following business requirements (Figure 4), it is further specified as: the number of days from when orders are ‘sent’ to when they are ‘closed’. The protocol model of the KPI ‘Percentage of In-Time Processed Orders’ should hence include some algorithm which, for a given time period, selects and counts only the instances of sent and closed orders for which this number of days equals or is lower than the number of days of the ‘Norm Lead Time’.

The sensitivity of ‘Processing Lead Time’ is specified as one day: only the changes in days of the ‘Processing Lead Time’ should be detected; those in seconds, minutes or hours should not be detected. Note that this refers to the sensitivity of the ‘Processing Lead Time’, but not of the KPI ‘Percentage of In-Time Processed Orders’; the value of the KPI changes only when the value of ‘Processing Lead Time’ of counted and selected instances of orders changes from being equal or lower than the ‘Norm Lead Time’ to being higher, or vice versa.

The formula for the KPI is specified in order to be able to guarantee congruency of the KPI value with the number of closed orders. The algorithm that is included in the protocol model should calculate the KPI value accordingly:

\[
\text{Percentage of In-Time Processed Orders} = \frac{\text{Closed (Norm)}}{\text{Closed}} \times 100\%
\]

\[
\text{Closed (Norm)} \text{ is the number of orders that are closed in a given time period (time end-time begin) and for which the ‘Processing Lead Time’ is equal or lower than the ‘Norm Lead Time’}
\]

\[
\text{Closed} \text{ is the number of closed orders in the given time period (time end-time begin).}
\]

A KPI is considered to have the property of reliability if no assumptions about the business process or the KPI formula are needed. KPI ‘Percentage of In-Time Processed Orders’ uses the number of orders, attributes-values, a given time interval, and the parameter ‘Norm Lead Time’. The number of orders and the values of the attributes are collected from the business process during the given time interval and do not need assumptions. However, it is unknown how the ‘Norm Lead Time’ is obtained: its definition needs assumptions and hence opens up possibilities to manipulate the KPI. Such possibilities for manipulation should be able to be confirmed with the protocol model (in the next section).

The property of improvement-orientation of KPI ‘Percentage of In-Time Processed Orders’ also needs further specification - although the KPI is derived from business strategy, related to a goal and does not use any measures of errors or violations. Further specification is needed of “improvement”. Hence, it is specified as an increase of the percentage of orders that is processed within the ‘Norm Lead Time’. It is assumed that such improvement cannot be manipulated in the business process, but this is not to say that there are no additional possibilities to manipulate. That is: all such possibilities should be taken into account.

Also a formula of performance measure ‘Average Processing Lead Time’ is specified:
Average Processing Lead Time
(Time End-Time Begin) = \[ \frac{\sum ('Processing Lead Time')}{Closed} \]

Closed is the number of closed orders in the given time period (time end-time begin).

Thus, the ‘Processing Lead Time’ is used for both the KPI ‘Percentage of In-Time Processed Order’ and the performance measure ‘Average Processing Lead Time’. Note however that when, for a given time period, the value of the latter equals or is lower than the ‘Norm Lead Time’, it is not necessarily so that the value of the former is “100%”. This is a significant but subtle difference between information provided by the KPI and by the performance measure.

5.3. Protocol Model

Following the above goals and requirements, a new protocol model is developed (Figure 5).

![Protocol Model Diagram]

Figure 5 Protocol model with a dashboard for ‘Percentage of In-Time Processed Orders’.

In the following the protocol model is explained. Some of the semantics are of particular interest, namely the inclusion of states and events in transitions, the use of data, and the CSP-composition of transitions. These are explained in greater detail using code that is written for the protocol model. Other semantics are only briefly addressed without code.
5.3.1. Sending and Confirming Orders

There is a protocol machine for the business concept *Order* which can be in the states *Sent* and *Confirmed* and can recognize the events *Send* and *Confirm*, among others (Figures 6 and 7). Actor *Logistical Employee* is responsible for the sending and confirming of orders.

<table>
<thead>
<tr>
<th>EVENT</th>
<th>ATTRIBUTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Send</td>
<td>Order: Order, \n</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EVENT</th>
<th>ATTRIBUTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Confirm</td>
<td>Order: Order</td>
</tr>
</tbody>
</table>

Figure 6 Part of the code of the protocol model that concerns the events *Send* and *Confirm*.

<table>
<thead>
<tr>
<th>OBJECT</th>
<th>Name</th>
<th>Includes</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order</td>
<td>Order Number</td>
<td>RegisterControl, \n</td>
<td></td>
</tr>
</tbody>
</table>

| STATES | Sent, Confirmed, Closed |

| TRANSITIONS | @new"Send=Sent", \n|            | Sent*Confirm=Confirmed, \n|            | Confirmed*Register=Confirmed, \n|            | Confirmed*ChangeInvoice=Confirmed, \n|            | Confirmed*Pay=Confirmed, \n|            | Confirmed*Close=Closed |

Figure 7 Part of the code of the protocol model that concerns the business concept of *Order*.

When event *Send* is presented from the environment and accepted by the protocol machine for *Order*, a new order-instance moves to state *Sent* following transition "@new"Send=Sent" (Figure 7). Before this transition is triggered, however, the following attributes of event *Send* must be specified (Figure 6): *Order Number*, which identifies the new order; *Order Amount*, which is the price of the goods that are ordered; and *Date Sent*, which is the date of sending the order. With event *Send* this data is communicated to the protocol machine *Order* which, after accepting the event, writes this data to its local storage as attributes of the new order.

When event *Confirm* is accepted, the order moves to state *Confirmed* following the transition “Sent*Confirm=Confirmed” (Figure 7). This is only possible for orders that are in state *Sent*, because the transition explicitly states that a move to the end-state *Confirmed* is triggered by event *Confirm* from this initial-state *Sent* (and there are no other transitions of the protocol machine for *Order*, or of any other protocol machine, in which the event *Confirm* is included).

5.3.2. Registering and Changing Invoices

There is a protocol machine for business concept *Invoice*, which can be in states *Registered* and *Paid* and can recognize events *Register* and *Change Invoice* (among others, Figures 8 and 9). Actor *Financial Employee* is responsible for registering and changing of invoices.
Figure 8 Part of the code of the model that concerns events Register and Change Invoice.

Figure 9 Part of the code of the protocol model that concerns the business concept Invoice.

When event Register is accepted by Invoice a new invoice moves to state Registered. Again some data must be specified, namely: Invoice Number, Invoice Amount, and Source. The latter identifies the order to which a new invoice relates. Several invoices might be related to a single source, but the source must be in state Confirmed. This is due to CSP-composition of transitions: event Register is included in both "@new*Register=Registered" of Invoice and "Confirmed*Register=Confirmed" of Order (Figure 7 and 9). These transitions occur only when the two protocol machines are in the required initial-states and accept event Register.5

With event Change Invoice the amount or source of an invoice can be changed. This is only possible for registered invoices, following "Registered*ChangeInvoice=Registered" of Invoice (Figure 9). There is CSP-composition with "Confirmed*ChangeInvoice=Confirmed" of Order (Figure 7), this time to enable the event Change Invoice to communicate the data on which orders are in state Confirmed and which can thus be specified as the (new) source of an invoice. Because this data is in the local storage of Order, this protocol machine should also recognize the event - and the event should hence be included in one of its transitions.

There are separate protocol machines for the behaviors of Register Control and of Change Control, which recognize the events Register and Change Invoice. Here CSP-composition is used for the modeling of a specific business rule, that is: Register Control accepts the events Register or Change Invoice if these events result in a state where the amount of a source is Not Exceeded by the total amount of invoices that are related to this source. If the amount is Exceeded, then the events should be (manually) refused. They might be accepted, but then Change Control moves to state Change Required and the event Change Invoice is desired.

5 Note that transition "Confirmed*Register=Confirmed" does not change the state of Order, because this is only to check whether an order to which an invoice relates is in the required state of Confirmed.
5.3.3. Paying Invoices and Closing Orders

Protocol machines *Invoice* and *Order* can also be in states *Paid* and *Closed* and recognize events *Pay* and *Close*, resp. (Figures 7, 9 and 10). Actor *Financial Employee* is responsible for the paying of invoices - and with the paying of invoices, orders are automatically closed.

```
EVENT ATTRIBUTES
Pay            Invoice: invoice,
                Source: order,
                Date Paid: date

EVENT ATTRIBUTES
Close          Order: order,
                Date Closed: date
```

Figure 10 Part of the code of the protocol model that concerns the events *Pay* and *Close*.

```
public void handleEvent() {
    Event pay = this.createEvent("Pay");
    pay.submitToModel();

    Instance myInvoice = this.getInstance("Invoice");
    Date datePaid = myInvoice.getDate("Date Paid");
    Instance myOrder = myInvoice.getInstance("Source");
    int orderAmount = myOrder.getCurrency("Order Amount");
    int orderPaidAmount = myOrder.getCurrency("Paid Amount");

    if (orderPaidAmount >= orderAmount) {
        Event close = this.createEvent("Close");
        close.getInstance("Order", myOrder);
        close.setDate("Date Closed", datePaid);
        close.submitToModel();
    }
}
```

Figure 11 Code of the callback-functions that are invoked when accepting the event of *Pay*.

When event *Pay* is accepted an invoice is *Paid* following “Registered!*Pay=Paid*" for *Invoice* (Figure 9 and 10), but also some callback-functions (marked “!”) are invoked for automatic processing (Figure 11). First, the invoice amount is added to the local storage of *Order* as attribute *Paid Amount*. Second, if the total *Paid Amount* at least equals the *Order Amount*,
then the source is automatically Closed with event Close, and Date Closed is written to the local storage of Order. Third, attribute Processing Lead Time of Order is calculated, namely as the difference in days between Date Closed and Date Sent in the local storage of Order.

A separate protocol machine for behavior Pay Control is used for the modeling of another specific business rule, that is: Pay Control accepts event Pay if this event results in a state where a source is Not Overpaid, i.e. if the order amount of a source is not exceeded by the total amount of paid invoices that are related to the source. If a source is Overpaid, then the event of Pay should be (manually) refused. It might be accepted, but this should be avoided.

5.3.4. Creating Dashboards

There is a protocol machine for the business concept Dashboard, which can be in the state Created and which can recognize the events Create Dashboard and Change Norm (Figures 12 and 13). This protocol machine is used for the modeling of the KPI ‘Percentage of In-Time Processed Orders’ of ‘Order to Pay’. The actor Business Manager is responsible for creating new dashboards.

![Figure 12](image)

Figure 12 Part of the code that concerns the events Create Dashboard and Change Norm.

![Figure 13](image)

Figure 13 Part of the code of the protocol model concerning business concept Dashboard.

With event Create Dashboard a new dashboard is created with these attributes: Dashboard Name, which is the identification of the dashboard; and Norm Lead Time, which is the norm (in days) for the ‘Processing Lead Time’ of the closed orders. Again, call-back functions are invoked (Figure 14). First, attribute Closed Order Count is calculated. Second, attribute Key Performance Indicator is calculated as the percentage of closed orders that is processed “in time”, i.e.: the count of orders that are ‘closed’ and for which the ‘Processing Lead Time’ equals or is lower than the ‘Norm Lead Time’ / the count of orders that are ‘closed’ * 100%.

---

6 Also included is attribute Average Processing Lead Time, which is the sum of the ‘Processing Lead Time’ of closed orders divided by their count. Note that when its value equals the ‘Norm Lead Time’, not necessarily all of the closed orders are processed “in time” (Key Performance Indicator = 100%).

23
In the above i is shown how KPI ‘Percentage of In-Time Processed Orders’ is specified and modeled using the conceptual model and a protocol model. It is also shown that its required properties imply that the ‘Processing Lead Time’ should be the (right) number of days from when orders are ‘sent’ to when they are ‘closed’, with a sensitivity of one day; and that the formula should be: “Percentage of In-Time Processed Orders (Time End - Time Begin) = Closed (Norm) / Closed * 100%’. A gap is identified with regard to specification of the ‘Norm Lead Time’, as it is unknown how it should be obtained. This should all be confirmed during validation.

5.4. Validation

With the protocol model the KPI ‘Percentage of In-Time Processed Orders’ and its required properties can be validated, that is: the data specifically on this KPI and its properties can be collected and analyzed. The results of the validation are summarized in the following, but the readers can and should be able to replicate the findings themselves (see Appendix B, for the code of the protocol model; and Appendix C, for the test-cases that are used while running the protocol model). This is not only to fully answer sub-question 4: How is the time-related KPI ‘Percentage of In-Time Processed Orders’ specified, modeled and validated?, but also to confirm the gap that was identified with regard to specification of the ‘Norm Lead Time’.

Quantifiability. KPIs should be measured through a process of quantification. The predefined functions that Protocol Modeling offers are used for selecting and counting the instances of orders in the model. For instance, the function “selectInState ("Order", “Closed”)” (Figure 14, under “getKeyPerformanceIndicator”) is used to select and count the orders that are closed as well as the closed orders that have a ‘Processing Lead Time’ that equals or is lower than the ‘Norm Lead Time’. The resulting values are stored as attributes and used for calculation of the KPI ‘Percentage of In-Time Processed Orders’. It is validated during the execution of
the protocol model that these functions result in the expected numbers and attribute-values.

**Sensitivity.** KPIs should be sensitive to changes in states and attribute-values. During execution the protocol model is populated by creating instances of orders and invoices in order to validate sensitivity of the KPI. By submitting different events for different instances, their states and attribute values change. It is validated during execution that these changes result in changes of the values of ‘Performance Lead Time’ and KPI ‘Percentage of In-Time Processed Orders’. Specifically, for instance, it is validated that the count of closed orders increases as more orders are closed and that the changes in days of ‘Processing Lead Time’ are detected, but not those in seconds, minutes or hours.

**Congruency.** KPIs should be congruent with changes in states and attribute-values. The congruency of the value of the KPI with the changes in states and attribute-values is tested during execution of the model. It shows, for instance, that small changes in the count of orders that are processed “in time” (i.e. orders for which the value of ‘Processing Lead Time’ equals or is lower than the given ‘Norm Lead Time’) lead to small changes in the value of the KPI. Furthermore, it shows that the values have a clear mathematical relationship (coded: “int KeyPerformanceIndicator = (int)((inTime*100.0f)/closedOrderCount”, Figure 14).

**Reliability.** KPIs should be semantically reliable and free from measurement error. It can be said that the KPI ‘Percentage of In-Time Processed Orders’ is free from errors in measurement resulting from subjective factors, because the value of the KPI is derived from (coded) states and attribute-values in the protocol model. Furthermore, the reliability can be validated during execution. Preferably this is done by business users, as these might have more knowledge of the KPI or the business process ‘Order to Pay’ and thus might be better able to uncover any incorrect or incomplete assumptions or specify additional business rules about the business process or the KPI formula.

**Improvement-Oriented.** KPIs should focus on improvement, not conformance. The KPI ‘Percentage of In-Time Processed Orders’ is derived from strategy, related to the ‘Norm Lead Time’ and does not use any measures of errors or violations. It can be shown during execution, however, that a business manager can freely set and change the ‘Norm Lead Time’ in conformance to (possibly manipulative) plans. Although this is in line with the above requirements, such a possibility to manipulate should be avoided. Additional business rules are needed, such as to have the ‘Norm Lead Time’ pre-coded, automatically derived from relevant data, or to have it set by an independent expert instead of a business manager on the basis of research and observation of the performance measure ‘Average Lead Time’. The properties of quantifiability, sensitivity and congruency are validated by creating instances of orders and invoices. Regarding the properties of reliability and improvement-orientation it is confirmed that there is at least one possibility to manipulate the KPI – by freely changing the ‘Norm Lead Time’, which demands the use of additional business rules.

6. Conclusion and Future Work

The aim of this research is to provide additional experimental evidence on how EXTREME can be used for the complete and correct specification, modeling and validation of KPIs, by replicating the results of earlier research, but with a new case study of the KPI ‘Percentage of In-Time Processed Orders’ - and while gaining insights into the properties of KPIs and the modeling semantics.

---

7 With regard to the property of efficiency it can be said that the code underlying the protocol model (Appendix B) shows that the KPI is easily quantifiable, feasible to collect and created in the simplest possible way out of the constituent measure ‘Processing Lead Time’. However, business users might validate that the KPI is easily understood and is not a waste of effort. Still, this is a subjective matter.
First, this work clarifies the definition of KPIs and their properties with a conceptual model of KPIs. The conceptual model shows that KPIs are complex and relate several concepts such as the business process and the KPI formula (which is based on elements of this business process). The conceptual model is notation-independent and may be used for specification of KPIs in any notation. The purpose of the conceptual model is twofold:

Classification of properties of KPIs on the basis of concepts used in the properties. Following the literature on performance management, a KPI is defined as “a metric used to quantify the efficiency and/or effectiveness of an action that is compared to a given norm or target which is derived from business strategy”. This definition gives a basic understanding, but KPIs should be specified, modeled and validated to have specific properties. In this study the recommendations on properties found in the literature are combined with a constructive definition of KPIs to build the conceptual model. The model explains how the properties can be grouped and represented by a shortlist of six required properties of KPIs that need to be taken into account. These properties are: quantifiability, sensitivity, congruency, reliability, efficiency and improvement-orientation.

Positioning of the case studies for the specification, modeling and validation of KPIs. The case studies in previous, related research are not time-related in the sense that they do not consider time norms for the business processes as border values in the KPI formula. In this work, therefore, a time-related KPI ‘Percentage of In-Time Processed Orders’ is used.

Second, this work replicates the method of modeling and validation of KPIs for the time-related process KPI ‘Percentage of In-Time Processed Orders’. This case study represents and covers the business processes that use time-related KPIs. In order to be able to ensure that a KPIs has the required properties, the modeling notation that is used should have semantics not only of business objects and their attributes, but also of states and events that are extended with data in order to make up transitions in the object-lifecycles of the business objects, and CSP-composition of these transitions to compose executable and deterministic behavior. The additional modeling semantics are not included in standard and popular modeling notations such as those in UML, which therefore are not suitable for the modeling and validation of KPIs. These modeling semantics are included in the method of EXTREME which combines ideas from goal modeling and Protocol Modeling.

KPI ‘Percentage of In-Time Processed Orders’ is initially specified as “the percentage of closed orders for which the ‘Processing Lead Time’ equals or is lower than the ‘Norm Lead Time’.” On the basis of the conceptual model and the (additional) modeling semantics, this is further specified. The ‘Processing Lead Time’ for instance is specified as “the number of days from when orders are ‘sent’ to when they are closed”. By using a protocol model, the KPI is validated to be calculated from: the right number of instances of business concepts (quantifiability); changes in states and attributes-values of the instances (sensitivity); a clear mathematical relationship (congruency); and a minimum number of assumptions (reliability) and scenarios for manipulation (improvement-orientation).

It has been found that the KPI might be improved, because the ‘Norm Lead Time’ can be freely changed and hence opens up possibilities to manipulate - additional business rules are needed. This shows that, if a KPI is not validated to have all of the required properties, the KPI might not reflect real performance or improvement and might be used to manipulate.

With this study new questions arise. Even if a KPI is validated to have all required properties, it might still not be a good KPI. “Goodhart’s law” for instance states, paraphrasing: “when a measure becomes a target, it ceases to be a good measure (Goodhart, 1975). This relates to the ‘observer effect’, i.e. that the process of measuring alone changes the material that is being measured. In this case this might mean, for instance, that the people participating in the business process change their behavior and performance; or that the manager in this case freely changes the ‘Norm Lead Time’ in order to manipulate the value of a KPI. How is
the targeting of a performance measure, the influencing or manipulation of a KPI prevented?

On the one hand the answer might lie with the involvement of operational participants in the specification, modeling and validation of a KPI, or in the separation of measurement and management. On the other hand the effect may be avoided for instance by keeping the performance measure secret, or through so-called “minimal interaction” or “habituation” (Saunders, Lewis & Thornhill, 2015). Future research is needed to answer such questions.
References


Appendix A: Relating Properties and Concepts

In the following the rationales are given for the relations in the conceptual model (Figure A).

Figure A Conceptual model of properties found in literature on performance management.

Relating the Shortlisted Properties to Concepts. The conceptual model (Figure A) is built by relating the shortlisted properties to concepts used in the constructive definition (Roubtsova & Michell, 2013a, 2013b, 2014) and to some additional concepts. This constructive definition is provided: “A KPI is a cumulative function of the cardinality of a set of selected business objects [from a business process] and the values of their attributes. The selected business objects give the value true to the selection predicate. The selection predicate compares the state and attribute values of each business object with the border values of attributes, the moment of selection and the time interval” (Roubtsova & Michell, 2013a, 2013b, 2014).

<table>
<thead>
<tr>
<th>Property: Concept</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantifiability: “Cardinality of […] business objects and the values of their attributes”</td>
<td>The cardinality is the number or quantity of the set of selected business objects.</td>
</tr>
<tr>
<td>Sensitivity: “KPI formula”</td>
<td>The KPI formula expresses the function of the cardinality.*</td>
</tr>
<tr>
<td>Congruency: “KPI Formula”</td>
<td>The KPI formula has both input and output for which there should be a (linear) mathematical relationship.*</td>
</tr>
<tr>
<td>Reliability: “Border values of attributes”</td>
<td>There should be no incorrect or incomplete assumptions on how to define the border values of attributes.</td>
</tr>
<tr>
<td>Improvement-orientation: ‘KPI value’, ‘Improvement’, ‘Goal’, ‘Strategy’ (additional)</td>
<td>Improvement is seen as a result of comparing the KPI values taken from at least two time periods. It is related to the goal that is derived from the business strategy.</td>
</tr>
<tr>
<td>Efficiency: ‘Management Reports’, ‘Managers’ (additional)</td>
<td>The management reports on the KPI values should be simple to understand for business managers and based on a count of the business objects and their attributes.</td>
</tr>
</tbody>
</table>

Table 1 Relating shortlisted properties to concepts from the definition or additional concepts.
Relating the Longlisted Properties to Concepts. The conceptual model (Figure A) is used for the categorization of properties by relating the longlisted properties found in literature to the concepts in the model. In order to do this the concepts and related shortlisted properties are grouped resulting in these following three groups: 1) “Quantifiability+Sensitivity+Congruency+Reliability”, 2) “Improvement-Oriented” and 3) “Efficiency”. Next, each longlisted property is placed into one of the three groups if the property uses some concepts of the group or if it forms a tautology with these concepts (it expresses the same concepts in a different way).

<table>
<thead>
<tr>
<th>#</th>
<th>Property: Group</th>
<th>Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Derived from strategy: Group 2</td>
<td>This refers to the concept of ‘Strategy’.</td>
</tr>
<tr>
<td>2</td>
<td>Simple to understand: Group 3</td>
<td>This refers to the 'Management Report' which should be simple to understand for the 'Manager' that uses it.</td>
</tr>
<tr>
<td>3</td>
<td>Timely and accurate feedback: Group 3</td>
<td>This refers to the 'Management Report' which should provide feedback for the 'Manager' that uses it.</td>
</tr>
<tr>
<td>4</td>
<td>Based on quantities that can be controlled: Group 1</td>
<td>This refers to the ‘Cardinality of [...] Business Objects and Values of their Attributes’, which are provided by the ‘Process’ and hence can be controlled.</td>
</tr>
<tr>
<td>5</td>
<td>Reflect the business process: Group 1</td>
<td>This refers to the 'Process' which provides the ‘Cardinality of [...] Business Objects and Values of their Attributes’.</td>
</tr>
<tr>
<td>6</td>
<td>Related to specific goals (targets): Group 1</td>
<td>This refers to the concept of ‘Goal’.</td>
</tr>
<tr>
<td>7</td>
<td>Relevant: Group 2</td>
<td>This is a different way to express that the ‘KPI Value’ and its ‘Improvement’ should relate to the ‘Goal’ that is derived from ‘Strategy’.</td>
</tr>
<tr>
<td>8</td>
<td>Part of management loop: Group 3</td>
<td>This refers to the 'Management Report' which should provide feedback for the 'Manager' that uses it.</td>
</tr>
<tr>
<td>9</td>
<td>Clearly defined: Group 1</td>
<td>This refers to the 'KPI Formula' and related concepts which should be clearly defined in order to be used.</td>
</tr>
<tr>
<td>10</td>
<td>Visual impact: Group 3</td>
<td>This refers to the 'Management Report' which should provide visual impact for the ‘Manager’ that uses it.</td>
</tr>
<tr>
<td>11</td>
<td>Focus on improvement: Group 2</td>
<td>This refers to the concept of ‘Improvement’.</td>
</tr>
<tr>
<td>12</td>
<td>Maintain significance, consistent in time: Group 1</td>
<td>This is a different way to express that the ‘KPI Value’ should be sensitive to the ‘Cardinality of [...] Business Objects and Values of their Attributes’.</td>
</tr>
<tr>
<td>13</td>
<td>Fast feedback: Group 3</td>
<td>This refers to the 'Management Report' which should provide fast feedback for the 'Manager' that uses it.</td>
</tr>
<tr>
<td>14</td>
<td>Have explicit purpose: Group 2</td>
<td>This is a different way to express that the ‘KPI Value’ and its ‘Improvement’ should relate to the ‘Goal’ that is derived from ‘Strategy’.</td>
</tr>
<tr>
<td>15</td>
<td>Based on explicitly defined formula and source of data: Group 1</td>
<td>This refers to the 'KPI Formula’ and the ‘Process’ (which provides the ‘Cardinality of [...] Business Objects and Values of their Attributes’), respectively.</td>
</tr>
<tr>
<td>16</td>
<td>Employ ratios instead of absolute numbers: Group 1</td>
<td>This refers to the 'KPI Formula' which should employ ratios and depends on the function-type and its coefficients.</td>
</tr>
<tr>
<td>17</td>
<td>Use data which are automatically extracted as part of a process: Group 1</td>
<td>This refers the ‘Cardinality of [...] Business Objects and Values of their Attributes’, the data on which is provided by the ‘Process’.</td>
</tr>
<tr>
<td>18</td>
<td>Reported in simple, consistent format: Group 3</td>
<td>This refers to the ‘Management Report’ which should have a simple and consistent format for the ‘Manager’ that uses it.</td>
</tr>
<tr>
<td></td>
<td>Based on trends rather than snapshots: Group 3</td>
<td>This refers to the 'Management Report' which should report the 'Improvement' of the 'KPI Value' based on trends.</td>
</tr>
<tr>
<td>---</td>
<td>-----------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>20</td>
<td>Provide information: Group 3</td>
<td>This refers to the 'Management Report' which should provide the feedback for the 'Manager' that uses it.</td>
</tr>
<tr>
<td>21</td>
<td>Precise (exact about what being measured): Group 3</td>
<td>This refers to the 'Management Report' which should be precise about the 'KPI Formula' and related concepts.</td>
</tr>
<tr>
<td>22</td>
<td>Objective (not based on opinion): Group 1</td>
<td>This refers the 'Cardinality of [...] Business Objects and Values of their Attributes' which should not be based on opinion but (automatically) provided by the 'Process'.</td>
</tr>
<tr>
<td>23</td>
<td>Easily measurable: Group 1</td>
<td>This refers the 'Cardinality of [...] Business Objects and Values of their Attributes' which should be easily measurable in order to be used in the 'KPI Formula'. Note that the emphasis is on the measurability, not on the ease of this (which would relate to the property of efficiency).</td>
</tr>
<tr>
<td>24</td>
<td>Inexpensive in time and cost: Group 3</td>
<td>This refers to the 'Management Report' which should be inexpensive in time and cost to be produced (which depends on for instance the 'Process' providing the 'Cardinality [...] and Values').</td>
</tr>
<tr>
<td>25</td>
<td>Sensitive to change: Group 1</td>
<td>This refers to the 'KPI Formula' which should be sensitive to the 'Cardinality of [...] Business Objects and Values of their Attributes'.</td>
</tr>
</tbody>
</table>

Table 1 Relating longlisted properties to concepts from the definition or additional concepts.
# Object Definitions

<table>
<thead>
<tr>
<th>OBJECT</th>
<th></th>
<th>NAME</th>
<th></th>
<th>INCLUDES</th>
<th></th>
<th>ATTRIBUTES</th>
<th></th>
<th>STATES</th>
<th></th>
<th>TRANSITIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order</td>
<td></td>
<td>Order Number</td>
<td></td>
<td>RegisterControl,</td>
<td>PayControl</td>
<td>Order Number: String,</td>
<td>Order Amount: Currency,</td>
<td>(Date Sent: Date),</td>
<td>(Date Closed: Date),</td>
<td>(!Processing Lead Time: Integer),</td>
</tr>
<tr>
<td>Invoice</td>
<td></td>
<td>Invoice Number</td>
<td></td>
<td>ChangeControl</td>
<td></td>
<td>Invoice Number: String,</td>
<td>Invoice Amount: Currency,</td>
<td>(Paid Amount: Currency),</td>
<td>Source: Order,</td>
<td>(Date Paid: Date),</td>
</tr>
<tr>
<td>!RegisterControl</td>
<td></td>
<td>ALLOWED</td>
<td></td>
<td>( !Registered Amount: Currency)</td>
<td></td>
<td>NotExceeded,</td>
<td>Exceeded</td>
<td>@any*Register=NotExceeded,</td>
<td>@any*ChangeInvoice=NotExceeded</td>
<td></td>
</tr>
<tr>
<td>!ChangeControl</td>
<td></td>
<td>DESIRED</td>
<td></td>
<td>ChangeRequired,</td>
<td>ChangeNotRequired</td>
<td>ChangeRequired*ChangeInvoice=ChangeRequired,</td>
<td>ChangeInvoice=@any</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**BEHAVIOUR**

**TYPE** | ALLOWED
---|---
**ATTRIBUTES** | (Paid Amount: Currency)
**STATES** | NotOverpaid, Overpaid
**TRANSITIONS** | @any*Pay=NotOverpaid

**OBJECT** Dashboard

**NAME** Dashboard Name

**ATTRIBUTES** Dashboard Name: String, Norm Lead Time: Integer, Average Processing Lead Time: Integer, Closed Order Count: Integer, Key Performance Indicator: Integer

**STATES** Created

**TRANSITIONS** @new*CreateDashboard=Created, Created*ChangeNorm=Created

---

# Event Definitions

**EVENT** Send

**ATTRIBUTES** Order: Order, Order Number: String, Order Amount: Currency, Date Sent: Date

**EVENT** Confirm

**ATTRIBUTES** Order: Order

**EVENT** Register

**ATTRIBUTES** Invoice: Invoice, Invoice Number: String, Invoice Amount: Currency, Source: Order

**EVENT** ChangeInvoice

**ATTRIBUTES** Invoice: Invoice, Invoice Amount: Currency, Source: Order

**EVENT** !Pay

**ATTRIBUTES** Invoice: Invoice, Source: Order, Date Paid: Date

**EVENT** Close

**ATTRIBUTES** Order: Order, Date Closed: Date

**EVENT** CreateDashboard

**ATTRIBUTES** Dashboard: Dashboard, Dashboard Name: String, Norm Lead Time: Integer

**EVENT** ChangeNorm

**ATTRIBUTES** Dashboard: Dashboard,
Norm Lead Time: Integer

# Actor Definitions

ACTOR Logistical Employee
BEHAVIOURS Order
EVENTS Send, Confirm

ACTOR Financial Employee
BEHAVIOURS Invoice
EVENTS Register, ChangeInvoice, Pay

ACTOR Business Manager
BEHAVIOURS Dashboard
EVENTS CreateDashboard, ChangeNorm

Callback Functions

package OrderToPay;
import com.metamaxim.modelscope.callbacks.*;
import java.util.*;
public class Order extends Behaviour {
    public int get________Amount() {
        int paidAmount = 0;
        Instance[] invoices = this.selectByRef("Invoice", "Source");
        for (int i = 0; i < invoices.length; i++)
            paidAmount += invoices[i].getCurrency("Paid Amount");
        return paidAmount;
    }
    public int get________Time() {
        Date DateClosed = this.getDate("Date Closed");
        Date DateSent = this.getDate("Date Sent");
        int ProcessingLeadTime = (int)( (DateClosed.getTime() - DateSent.getTime()) / (1000 * 60 * 60 * 24) )
        return ProcessingLeadTime;
    }
    public String get________State() {
        return this.getState("Order");
    }
}

package OrderToPay;
import com.metamaxim.modelscope.callbacks.*;
import java.text.DecimalFormat;
public class OrderInit {
    public static String nextOrderNo (Instance anyThing) {
        Instance[] orders = anyThing.selectInState("Order", "@any");
        return(new DecimalFormat("000").format(orders.length + 1));
    }
    public static int defaultOrderAmount() {
        return 1000;
    }
}

package OrderToPay;
import com.metamaxim.modelscope.callbacks.*;
import java.util.*;
public class Invoice extends Behaviour {
    public void processPay(Event event, String subscript) {
    
    
    
    }

package OrderToPay;
import com.metamaxim.modelscope.callbacks.*;
import java.util.*;
public class OrderInit {
    public static String nextOrderNo (Instance anyThing) {
        Instance[] orders = anyThing.selectInState("Order", "@any");
        return(new DecimalFormat("000").format(orders.length + 1));
    }
    public static int defaultOrderAmount() {
        return 1000;
    }
}

package OrderToPay;
import com.metamaxim.modelscope.callbacks.*;
import java.util.*;
public class Invoice extends Behaviour {
    public void processPay(Event event, String subscript) {
        // Implementation
    }
}
int PaidAmount = this.getCurrency("Invoice Amount");
this.setCurrency("Paid Amount", PaidAmount);
public String getCurrentState() {
return this.getState("Invoice"); }

package OrderToPay;
import com.metamaxim.modelscope.callbacks.*;
import java.text.DecimalFormat;
public class InvoiceInit {
public static String nextInvoiceNo (Instance anyThing) {
Instance[] invoices = anyThing.selectInState("Invoice", "@any");
return(new DecimalFormat("000").format(invoices.length + 1));
public static int defaultInvoiceAmount() {
return 1000; }

package OrderToPay;
import com.metamaxim.modelscope.callbacks.*;
public class RegisterControl extends Behaviour {
public int getRegisteredAmount() {
int registeredAmount = 0;
Instance[] invoices = this.selectByRef("Invoice", "Source");
for (int i = 0; i < invoices.length; i++)
registeredAmount += invoices[i].getCurrency("Invoice Amount");
return registeredAmount;
public String getState() {
if (this.getCurrency("Registered Amount") >
this.getCurrency("Order Amount"))
return "Exceeded";
else return "NotExceeded"; }

package OrderToPay;
import com.metamaxim.modelscope.callbacks.*;
public class ChangeControl extends Behaviour {
public String getState() {
String regState = this.getInstance("Source").getState("RegisterControl");
if (regState.equals("Exceeded"))
return "ChangeRequired";
else return "ChangeNotRequired"; }

package OrderToPay;
import com.metamaxim.modelscope.callbacks.*;
public class PayControl extends Behaviour {
public int getPaidAmount() {
int paidAmount = 0;
Instance[] invoices = this.selectByRef("Invoice", "Source");
for (int i = 0; i < invoices.length; i++)
paidAmount += invoices[i].getCurrency("Invoice Amount");
return paidAmount;
public String getState() {
if (this.getCurrency("Paid Amount") >
this.getCurrency("Order Amount"))
return "Overpaid";
else return "NotOverpaid"; }

package OrderToPay;
package OrderToPay;
import com.metamaxim.modelscope.callbacks.*;
public class DashboardInit {
  public static String defaultDashboardName() {
    return "KPI In-time Processing";
  }
}

package OrderToPay;
import com.metamaxim.modelscope.callbacks.*;
public class Send extends Event {
  public void setOrderNumber(EventValueAttribute attribute, Instance selected, String subscript) {
    attribute.setString(OrderInit.nextOrderNo(selected));
  }
  public void setOrderAmount(EventValueAttribute attribute, Instance selected, String subscript) {
    attribute.setCurrency(OrderInit.defaultOrderAmount());
  }
}

package OrderToPay;
import com.metamaxim.modelscope.callbacks.*;
public class Register extends Event {
  public void setInvoiceNumber(EventValueAttribute attribute, Instance selected, String subscript) {
    attribute.setString(InvoiceInit.nextInvoiceNo(selected));
  }
  public void setInvoiceAmount(EventValueAttribute attribute, Instance selected, String subscript) {
    attribute.setCurrency(InvoiceInit.defaultInvoiceAmount());
  }
}
public class Pay extends Event {
    public void handleEvent() {
        Event pay = this.createEvent("Pay");
        pay.submitToModel();
        Instance myInvoice = this.getInstance("Invoice");
        Date datePaid = myInvoice.getDate("Date Paid");
        Instance myOrder = myInvoice.getInstance("Source");
        int orderAmount = myOrder.getCurrency("Order Amount");
        int orderPaidAmount = myOrder.getCurrency("Paid Amount");
        if (orderPaidAmount >= orderAmount) {
            Event close = this.createEvent("Close");
            close.setInstance("Order", myOrder);
            close.setDate("Date Closed", datePaid);
            close.submitToModel();
        }
    }
}

import com.metamaxim.modelscope.callbacks.*;
import java.util.*;

package OrderToPay;
import com.metamaxim.modelscope.callbacks.*;
public class CreateDashboard extends Event {
    public void setDashboardName(EventValueAttribute attribute, Instance selected, String subscript) {
        attribute.setString(DashboardInit.defaultDashboardName());
        attribute.setRule("WORD_CHAR_RULE", "Dashboard Name must be supplied");
    }
}

package OrderToPay;
import com.metamaxim.modelscope.callbacks.*;
public class CreateDashboard extends Event {
    public void setDashboardName(EventValueAttribute attribute, Instance selected, String subscript) {
        attribute.setString(DashboardInit.defaultDashboardName());
        attribute.setRule("WORD_CHAR_RULE", "Dashboard Name must be supplied");
    }
}
Appendix C: Test-Cases used during Validation

# Metamaxim ModelScope Instances File

INSTANCE : KPI In-time Processing = 1
    BEHAVIOUR : Dashboard = Created
        Dashboard Name : String = KPI In-time Processing
        Norm Lead Time : Integer = 3

INSTANCE : 001 = 2
    BEHAVIOUR : Invoice = Paid
        Invoice Number : String = 001
        Invoice Amount : Currency = 10.00
        Paid Amount : Currency = 10.00
        Source : Order = 5
        Date Paid : Date = 19 Apr 2016
    BEHAVIOUR : ChangeControl = ChangeNotRequired

INSTANCE : 002 = 3
    BEHAVIOUR : Invoice = Paid
        Invoice Number : String = 002
        Invoice Amount : Currency = 10.00
        Paid Amount : Currency = 10.00
        Source : Order = 6
        Date Paid : Date = 21 Apr 2016
    BEHAVIOUR : ChangeControl = ChangeNotRequired

INSTANCE : 003 = 4
    BEHAVIOUR : Invoice = Registered
        Invoice Number : String = 003
        Invoice Amount : Currency = 12.00
        Paid Amount : Currency = 0.00
        Source : Order = 7
        Date Paid : Date = 17 Apr 2016
    BEHAVIOUR : ChangeControl = ChangeNotRequired

INSTANCE : 001 = 5
    BEHAVIOUR : Order = Closed
        Order Number : String = 001
        Order Amount : Currency = 10.00
        Date Sent : Date = 17 Apr 2016
        Date Closed : Date = 19 Apr 2016
    BEHAVIOUR : RegisterControl = NotExceeded
    BEHAVIOUR : PayControl = NotOverpaid

INSTANCE : 002 = 6
    BEHAVIOUR : Order = Closed
        Order Number : String = 002
        Order Amount : Currency = 10.00
        Date Sent : Date = 17 Apr 2016
        Date Closed : Date = 21 Apr 2016
    BEHAVIOUR : RegisterControl = NotExceeded
    BEHAVIOUR : PayControl = NotOverpaid

INSTANCE : 003 = 7
    BEHAVIOUR : Order = Confirmed
        Order Number : String = 003
Order Amount: Currency = 10.00
Date Sent: Date = 17 Apr 2016
Date Closed: Date = 17 Apr 2016
BEHAVIOUR: RegisterControl = Exceeded
BEHAVIOUR: PayControl = Overpaid

INSTANCE: 004 = 8
BEHAVIOUR: Order = Sent
    Order Number: String = 004
    Order Amount: Currency = 10.00
    Date Sent: Date = 18 Apr 2016
    Date Closed: Date = 18 Apr 2016
BEHAVIOUR: RegisterControl = NotExceeded
BEHAVIOUR: PayControl = NotOverpaid