Towards Compliance Checking between Business Process Models and Lawful States of Objects

Ludmila Penicina and Marite Kirikova

Institute of Applied Computer Systems, Riga Technical University, 1 Kalku, Riga,
LV-1658, Latvia
`{ludmila.penicina, marite.kirikova}@rtu.lv`

Abstract. We address the existing gap between business process models and lawful states of business objects. This gap hinders compliance of business process models with internally and externally imposed regulations. Existing modelling methods such as BPMN and ArchiMate lack an explicitly declarative approach for capturing flow of business objects, their states and laws of state transitions. Such deficiency can cost organization potential legal problems, make the ability of BPMN and ArchiMate to capture real-world phenomena questionable and drive modellers to employ additional standards. This paper proposes a formalized solution for closing the gap between business process models and states of business objects by using BWW model. Our approach includes means for explicit definition of states of business objects, automatic generation of conceivable state space at a process model design-time, and automatic generation of lawful state space and compliance checking at a process run-time.

Keywords: Business process modelling, BWW, BPMN, Object state, Compliance.

1 Introduction

Business processes are valuable assets of any organization. In organizations business process modelling has become a main activity for capturing, analysing, and improving business processes. Business process modelling comprises two aspects – the control-flow perspective and data-flow perspective [1]. Control-flow perspective defines possible execution paths of a business process, while data-flow perspective represents how business objects are manipulated and change states during a process. Data in business process models are usually declared in terms of business objects (physical or virtual) and usually there are prescribed allowed states of business objects contained in internal business policies, external legislative documents, standards, reference models, and other regulations. Nowadays there is an increased pressure on organizations to guarantee compliance of their business processes with various regulatory and legislative requirements, other externally imposed constraints, and internal business policies [2]. For an organization engaged in business process modelling this might mean that (1) activities in business process models have to be associated with business objects representing inputs or outputs, (2) it has to be possible to represent a state of a

business object at a given point of time, (3) it has to be possible to associate allowed state transitions with a business process model, and (4) it has to be possible to detect if a state of a business object is compliant with allowed state transitions. In this paper we are not talking about the soundness of the process – correctness criteria that a process model has to fulfil, e.g., deadlock or livelock patterns.

Compliance can be checked during or after the execution of the business process, called *compliance by detection*, or compliance can be checked while modelling the business process, called *compliance by design* [3]. In this paper we address the issue of compliance between business process models and lawful state space of business objects. In our solution we intend to apply *compliance by detection* method to check during the execution of the business process if states of business objects are compliant with the lawful state space. However, we also intend to generate a space of conceivable states for business objects at a design-time of business process.

We motivate our research with the following: compliance between business process models and lawful state space of business objects (1) ensures that organization will not violate laws and there will be no potential legal problems for the organization, and (2) ensures consistency in collaborative business processes and customer satisfaction. A number of studies exist that show the importance of addressing data and states of data in business process models – e.g., in [4] authors indicate the importance of *data-driven* process structures in large engineering processes such as assembling of a car or an airplane, and according to [5] in order to achieve safe execution of a process model it must be ensured that every time a task attempts to access a data object, the data object is in a certain expected data state (legal state). And since not all possible transitions of states are meaningful, restrictions on object state transitions are also required. In this paper we intentionally use the term "business objects" and not "data objects", since active structure elements are also capable of assuming a state which can be illegal and should be also monitored.

Nowadays organizations employ industry modelling standards like BPMN [6] and ArchiMate [7] to understand and improve business processes. Business Process Model and Notation (BPMN) [6] is the de-facto standard for representing in a very expressive graphical way the processes occurring in virtually every kind of organizations [8]. However, BPMN has its limitations when it comes to modelling other aspects of organizations such as organizational structure and roles, functional breakdowns, data, strategy, business rules and technical systems [9]. Information about Enterprise Architecture (EA) is needed to create real-world business process models. To provide a uniform representation for diagrams that describe EA, ArchiMate modelling language has been developed [7]. The core of ArchiMate language consists of three main types of elements: active structure elements, behaviour elements, and passive structure elements (objects) [7]. Some tools like ARIS [10] and QPR [11] allow linking BPMN and ArchiMate models in their modelling environments. Linkage between BPMN models and ArchiMate models provides possibilities to complement BPMN models with enterprise aspects and ArchiMate models with detailed process descriptions. In this paper we particularly address linked BPMN and ArchiMate models, which we, for simplicity reasons, call *business process models*.

The previous research has shown (see [12], [13] and [14]) that BPMN and ArchiMate lack in ability to describe flow of business objects in business process models and explicitly declare states of business objects imposed by regulations. This

gap hinders compliance of business process models with external and internal regulations.

Wand and Weber [15] built a set of models for the evaluation of modelling techniques based on an upper ontology defined by Bunge [16]. They extended Bunge's ontology and applied it to the modelling of information systems (BWW model) [15]. BWW model consists of constructs present in the real world that must be represented in information systems. BWW model allows straightforwardly addressing: (1) states of things, (2) lawful state space and lawful event space of things, (3) conceivable state space and conceivable event space of things, (4) state law that restricts values of the properties of things to a lawful subset, and (5) lawful transformations that define which events in things are lawful. To be able to control whether an unlawful event has occurred in a business process, or a business object has assumed an unlawful state, it is necessary: (1) to provide means explicitly defining states of business objects in business process models (2) to generate lawful and conceivable states spaces for business process models, and (3) to check compliance of business process models with generated lawful state spaces at a run-time.

This paper presents an on-going research which aims to provide a solution and a prototype of a tool for supporting explicit declaration of lawful states and compliance checking between business process models and lawful state space of business objects. For a theoretical foundation purpose we propose to use BWW model [15], since BWW model complements BPMN and ArchiMate for what they are lacking – explicit representation of business objects, their states, and state transition laws.

Research presented in [17] describes how BPMN and ArchiMate support BWW model. There are 6 BWW model elements that are not supported by these modelling languages, namely, *State Law (SL), Conceivable State Space (CSS), Lawful State Space (LSS), History (H), Conceivable Event Space (CES),* and *Lawful Event Space (LES)*: or a tuple {SL, CSS, LSS, H, CES, LES}. These six elements are to be taken into consideration to define a complete, lawful, and consistent description of business processes. Our work focuses on the use of BWW elements {SL, CSS, LSS, H, CES, LES} in designing compliant with the states of business objects business process models. However, we are aware that the subject of compliance is broader than concerns of business object states.

The main contribution of this paper resides in that we use BWW model – a system's model with a proven research record – to supplement BPMN and ArchiMate models with explicit declarations of object states, state laws and conceivable and lawful state spaces in order to support organizations in achieving compliance with regulations.

The paper is structured as follows. In Section 2 the related work is outlined. In Section 3 a running example that we use throughout the paper is described. Section 4 contains formalization of BWW elements {SL, CSS, LSS, H, CES, LES} using a set theory. In Section 5 existing gaps and the proposed solution is discussed. Brief conclusions and future work are presented in Section 6.

2   Related Works

The lack of consistent theoretical foundation for building information systems urged Wand and Weber [15] to build a set of models for the evaluation of modelling

techniques. Wand and Weber have extended the ontology presented by Mario Bunge [16] and developed a formal foundation called BWW model for modelling information systems [15]. Elements in BWW model (in the text shown in italics) can be organized in the following groups (adapted from [17]):

1. *Thing* – including *Properties, Classes* and *Kinds* of *Things*. *Thing* is an elementary unit in BWW. *Things* possess *Properties*, which defines *States* of a *Thing*. *Things* can belong to *Classes* or *Kinds* depending on a number of common *Properties*. A *Thing* can act on another *Thing* if its existence affects the *History* of the other *Thing*. *Things* are coupled if one *Thing* acts on another.

2. *State of Thing* – *Properties* of *Things* define their *States*. *State Law* restricts *Values* of *Properties* of *Things*. *Conceivable State Space* is a set of all *States* a *Thing* can assume. *Lawful State Space* defines *States* that comply with *State Law*. *Stable State* is a *State* in which *Thing* or a *System* will remain unless forced to change by *Thing* in the *System Environment*. *Unstable State* is *State* that will be changed into another *State* by the *Transformations* in the *System*. *History* is the chronologically-ordered *States* of *Thing*.

3. *Transformation* – transformation between *States of Things*. *Transformation* is a mapping from one *State* to another. *Lawful Transformation* defines which *Events* in *Thing* are lawful.

4. *Event* – is a change in *State* of *Thing*. *Conceivable Event Space* is a set of all *Events* that can occur to *Thing*. *Lawful Event Space* is a set of all *Events* that are lawful to *Thing*. *Events* can be *Internal Events* and *External Events*. *Events* can be *Well-Defined* – *Event* in which the subsequent *State* can be predicted – or *Poorly-Defined Event* in which the subsequent *State* cannot be predicted.

5. *System* – a set of coupled *Things*. *System Composition* is *Things* in the *System*. *System Environment* is *Things* outside the *System* interacting with the *System*. *System Structure* is a set of couplings that exist among *Things*. *Subsystem* is *System* whose composition and structure is a subset of the composition and structure of another *System*. *System Decomposition* is a set of *Subsystems*. *Level Structure* is an alignment of the subsystems.

The authors of [5] propose a notion of "weak conformance" which checks conformance of a process model with respect to data objects. This notion can be used to tell whether in every execution of a process model each time a task needs to access a data object in a particular state, it is ensured that the data object is in the expected state or can reach the expected state and, hence, the process model can achieve its goals. In [18] authors identify that consistency between business process models and object life cycle is required, however, their relation is not well understood. Authors clarify this relation and propose an approach to establish the required consistency by explicitly defining object states in business process models and then generating life cycles for each object type in the process. The authors of [18] indicate that object life cycle modelling is valuable at the business level. However, we propose to consider states of objects also at the application and technology levels of enterprise architecture since objects can be hidden and specified in sub-process structures at different levels of an enterprise. The authors of [19] use object life cycle as a common means for explicitly modelling allowed state transitions of an object during its existence and propose a technique for generating a compliant business process model from a set of given reference object life cycles.

The notion of a "legal state" is also mentioned in [20] where authors indicate that the representation of legal states in a model of a trade procedure is essential because organizations should be able to derive their obligations, rights, and duties at each point during the execution of the trade procedure and propose to annotate the states in Petri nets. In [2] authors investigate the use of temporal deontic assignments on activities as a means to declaratively capture the control-flow semantics that reside in business regulations and business policies. In object-oriented paradigm, state machines are extensively used for representation of states of objects [21]. In [22] the authors propose logic based formalism for describing the semantics of business contracts and the semantics of compliance checking procedures and close the gap between business processes and business contracts. In [3] the author focuses on compliance by design and extends artifact-centric approach to model compliance rules using Petri nets and show how compliant business processes can be synthesized automatically from the point of view of the involved business objects.

Since we address the importance of explicitly representing business objects and their states in business process models, our approach is also related to case handling [23] – a relatively new paradigm that, unlike workflow management, is strongly based on data. In our approach we generate a lawful state space using a conceivable state space based on a particular business process scenario (case).

The objective of this paper differs from the related work in that it uses BWW model as a theoretical foundation for generating conceivable and lawful state spaces from a business process model and applies it to nowadays de-facto modelling methods BPMN and ArchiMate.

3   Example: Electronic Submission

Throughout this paper we are using a simple electronic submission example at a university in which a researcher uploads his publication to university repository and can choose an option to publish her work as Open Access publication (see Figure 1). Researchers must choose a licence under which they wish to publish their publication – a version of the full text of the work which the publisher permits to archive in the institutional repository. The possible versions of the publication's full texts are: pre-print, post-print or published version. Uploaded publication can assume several states based on the set of its properties, e.g., lawful state will be when a version of a publication's full text is the pre-print and publisher has permitted archiving this publication. Lawful event will be allowing showing a full text of this publication publicly. Unlawful event will be when a publisher has not allowed archiving but a full text is made available publicly.
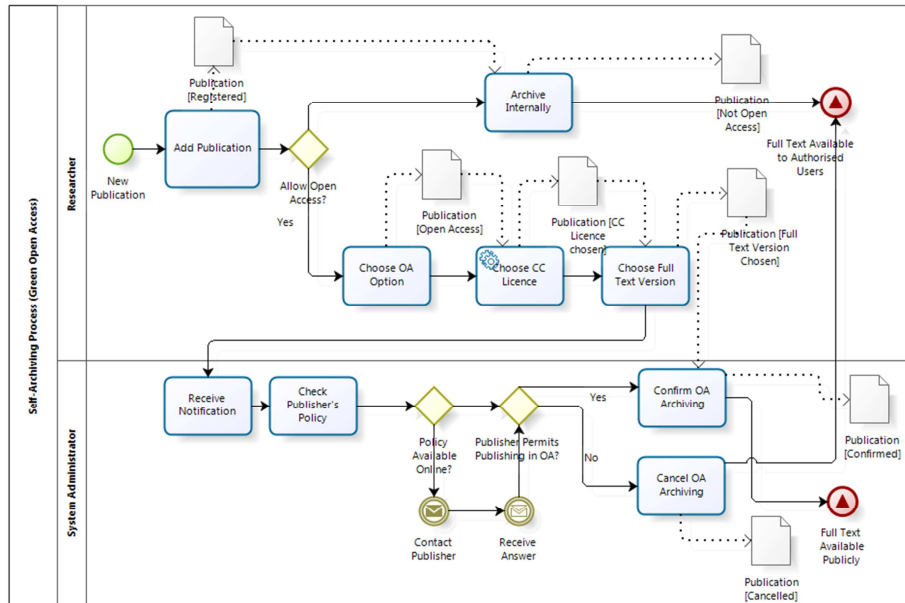
Fig. 1. BPMN 2.0 model of the electronic submission business process.

4   Formalization of BWW Model

In this section we propose formal definitions of BWW model elements based on informal description of BWW model presented in [12].

Definition 1: *Thing*. *A Thing is the elementary unit in the BWW ontological model. The real world is made up of Things. Things possess Properties.*
A *Thing* is a tuple:

$$T = \{P, SL, CSS, LSS, H, LT, CES, LES\}, \text{ where:}$$

- P is a set of *Properties* of a *Thing*
- SL is a *State Law* of a *Thing*
- CSS is a *Conceivable State Space* of a *Thing*
- LSS is a *Lawful State Space* of a *Thing*
- H is a *History* of a *Thing*
- LT is *Lawful Transformation* of a *Thing*
- CES is a *Conceivable Event Space* of a *Thing*
- LES is a *Lawful Event Space* of a *Thing*

Example. In the running example presented in Section 3 *Thing* is a Publication submitted by a Researcher.

Definition 2: *Property*. *A Property is modelled via a function that maps the Thing into some value.*
Property is a tuple:

$$P = \{a, t\}, \text{ where:}$$

- a is an *Attribute* of a *Property*
- t is a *Property* type, namely, in general, in particular, hereditary, emergent, intrinsic.

Property is described as a function that maps a *Thing* from a set of *Properties* P_x to P_y:

$$(f : P_x \rightarrow P_y).$$

Example. In the running example presented in Section 3 Publication is assumed to have the following *Properties* (due to limitation of space we present only a subset of all possible properties):

- P_1={Title, In General}.
- P_2={Status, In General} – differs from the notion *State* (although names can be identical). Values of "Status" can be "Registered, "Confirmed", "Cancelled".
- P_3={Open Access Mark, In General} – represents whether a Researcher has chosen the option to archive Publication as Open Access.
- P_4={CC Licence, In General} – represent chosen CC License, possible values: "CC BY", "CC BY-SA", "CC BY-ND".
- P_5={Version of the Full Text, In General} – can have values "pre-print" "post-print", or "publisher's version/PDF".
- P_6={Publisher Policy, In General} – can have values "Green" (can archive pre-print and post-print or publisher's version/PDF), "Yellow" (can archive pre-print), "White" (archiving not formally supported).

Definition 3: _State_. *The vector of values for all Property functions of a Thing.*

Let's assume that there is Publication X, then a *State* for a Publication X at a given point of time can be defined as

$$S_{xi} = \{ID, \{P_1, P_2,...P_i, P_{i+1},...P_n\}\}, \text{ where:}$$

- ID is a name that identifies the *State*
- $\{P_1, P_2,...P_i, P_{i+1},...P_n\}$ is the vector of values for all *Property* functions

Example. *State* for a Publication X from the running example:

S_{Px} = {Confirmed, {Title X, Confirmed, Yes, CC BY, Pre-Print, Yellow}}

Definition 4: _Conceivable State Space_. *The set of all States that the Thing might ever assume.*

$$CSS = \{S, T\}, \text{ where:}$$

- S is a set of finite conceivable *States*
- T is a *Transformation* that is a mapping function, e.g., from *State* X to *State* Y: $(f_t: S_x \rightarrow S_y)$ – it is an association to a particular activity in the business process model.

Example. For any uploaded Publication X from our running example:

CSS_P = {{Registered, Add Publication}, {Open Access, Choose OA Option}, {Not Open Access, Archive Internally}, {CC Licence Chosen, Choose CC Licence}, {Full Text Version Chosen, Choose Full Text Version}, {Publication Confirmed, Confirm OA Archiving}, {Publication Cancelled, Cancel OA Archiving}}

Definition 5: _State Law_. *A State Law restricts the values of the Properties of a Thing to a subset that is deemed lawful.*

$$SL = \{P_{law}\}, \text{ where:}$$

P_{law} are *Properties* of a *Thing* that are lawful and is a subset of *Properties* of a *Thing*:

$$P_{law} \sqsubseteq P$$

Example. In the electronic submission example a *State* "Full Text Available Publicly" is lawful only in case when *Properties* of Publication are, .e.g.:

- $P_1=\{\text{Title="Title X", In General}\}$
- $P_2=\{\text{Status= "Confirmed", In General}\}$
- $P_3=\{\text{Open Access Mark= "Yes", In General}\}$
- $P_4=\{\text{CC Licence="CC BY", In General}\}$
- $P_5=\{\text{Version of the Full Text= "Pre-Print", In General}\}$
- $P_6=\{\text{Publisher Policy = "Yellow", In General}\}$

Definition 6: *Lawful State Space*. *The set of States of a Thing that comply with State Laws of the Thing.*

$$LSS = \{S, SL\} \text{ where:}$$

- S is a set of finite lawful *States*
- SL is a *State Law* – set of Properties that are lawful for a Thing in this particular state

Example. Let's assume that a Researcher has uploaded a particular Publication X, then:

$LSS_{Px} = \{\{$Registered, {Title X, Registered}}, {Open Access, {Title X, Registered, Yes}}, {CC Licence Chosen, {Title X, Registered, Yes, CC BY}}, {Full Text Version Chosen, {Title X, Registered, Yes, CC BY, Pre-Print, Yellow}}, {Publication Confirmed, {Title X, Confirmed, Yes, CC BY, Pre-Print, Yellow }}

Definition 7: *History*. *The chronologically ordered states that a Thing traverses in time.*

$$H = \{s_s, s_i, \ldots, s_n, \ldots s_e\}, \text{ where:}$$

- s_s is a start *State*
- s_i and s_n are chronologically next *States* in time
- s_e is an end *State*

Example. History of *States* in the running example for a Publication X:

$H_{Px} = \{$Registered, Open Access, CC Licence Chosen, Full Text Version Chosen, Confirmed}

Definition 8: *Lawful Transformation*. *Defines which Events in a Thing are lawful. Event is a change in a State of a Thing.*

$$LT = \{E_l, SC, CA\}, \text{ where:}$$

- E_l is a set of *Events* that are lawful in a *Thing*, it can be defined as a subset of all *Events:* $E_l \sqsubseteq E$
- SC is a set of *Stability Conditions* that specify the *States* that are lawful under *Lawful Transformation*
- CA is a set of *Corrective Actions* that specify how the values of the *Property* functions must change to provide a *State* acceptable under transformation law.

$$CA=\{(f:Px \rightarrow Py)\}$$

Example. In the running example LT for a Publication X in a *State* "Registered":

LT$_{Px}$={{E$_1$={Registered→Open Access}, SC$_{E1}$ ={Registered, Open Access}, CA $_{E1}$ ={{Title X, Registered}→{Title X, Registered, Yes}}}, {E$_2$={Registered→Not Open Access}, SC$_{E1}$ ={Registered, Not Open Access}, CA $_{E1}$ ={{Title X, Registered}→{Title X, Registered, No}}}}}

Definition 9: ***Conceivable Event Space***. *The set of all possible Events that can occur in the Thing.*

$$CES = \{E, T\}, \text{ where:}$$

- E is a set of all Events that can occur in a *Thing*
- T is a *Transformation* that is a mapping function, e.g., from *State* X to *State* Y: *(f$_t$: S$_x$→S$_y$)*

Example. In the running example CES for Publication X:

CES$_{Px}$ ={{E$_1${Registered→Open Access}, E$_2${Registered→Not Open Access}, E$_3${Open Access→CC Licence Chosen}, E$_4${CC Licence Chosen→Full Text Version Chosen}, E$_5${Full Text Version Chosen→Publication Confirmed}, E$_6${Full Text Version Chosen→Publication Cancelled}

Definition 10: ***Lawful Event Space***. *The set of all Events in a Thing that are lawful.*

$$LES = \{E_l, LT\} \text{ where:}$$

- E is a set of lawful *Events*
- LT is a *Lawful Transformation*

Example. In the running example LES for Publication X:

LES$_{Px}$={E$_1${Registered→Open Access}, E$_2${Open Access→CC Licence Chosen}, E$_3${CC Licence Chosen→Full Text Version Chosen}, E$_4${Full Text Version Chosen→Publication Confirmed}}.

The applications of above-presented formalizations will be shown in Section 5.

5 Existing Gaps and Proposed Solution

This paper continues the research presented in [14] and [13] where the evaluation of BPMN and ArchiMate against BWW was presented. Based on the results presented in previous works, we can conclude that BWW model defines a set of elements that are supported by BPMN and ArchiMate modelling language as well as a set of elements that are not supported by these modelling languages. Majority of BPMN and ArchiMate core elements can be mapped to BWW constructs. However, it is necessary to supplement BPMN and ArchiMate modelling languages with the missing elements in order to be able to maintain a set of lawful object states in business process models.

Because in BPMN and ArchiMate there is no explicit representation for object's *State*, *Conceivable State Space*, *Lawful State Space, State Law, Conceivable Event Space, Lawful Event Space,* and *History* – resulting BPMN and ArchiMate models may be irrelevant and modellers may need to incorporate additional modelling techniques to overcome these defects. It may be impossible to detect from BPMN and ArchiMate models which events and states should be expected to occur and which events and states can occur but are illegal. Another important aspect is lacking of element *History*,

which chronologically describes state changes of business objects. This deficiency can lead to problems regarding maintaining system's log and recovery.

These gaps hinder lawfulness of business process models, because lawful states of business objects are not explicitly depicted in business process models, models might contain meaningless states and events, since a set of conceivable states and events are not depicted, and, as a result, business process models do not represent real-world processes and can lead to business process incompliance with regulations. Also, since BPMN proclaims to be directly executable, omitting states and state transition laws may hinder correct automated execution.

Using BWW model will potentially support creating business process models compliant with regulations, since missing BWW elements are addressed. Our approach intends to achieve the following:

- Explicitly defining *Properties* of business objects in business process models using formal definition described in Section 4 and indicating whether business object is an input or output parameter of an activity.
- Explicitly defining *States* of business objects in business process models using formal definition described in Section 4.
- At business process design-time we intend to generate automatically *State Law, Conceivable State Space* and *Conceivable Event Space* directed graphs based on formal definitions presented in Section 4 and explicitly defined *Properties* and *States* of business objects.
- We intend to check compliance of business process with lawful states of business objects at a run-time. At business process run-time based on a particular process scenario or case, we intend to generate automatically *Lawful State Space, History,* and *Lawful Event Space* directed graphs.
- We intend to use rules for object life cycle generation presented in [18] for automatically generating conceivable and lawful state spaces. Rules for object life cycle generation presented in [18] are based on patterns that are matched in the business process model and used to create object life cycle with state transitions from initial state to possible end states.

The proposed solution for maintaining lawful states of business objects in business process models requires a repository-based modelling tool that accommodates BPMN, ArchiMate and BWW.

For the running example of electronic submission of a research paper to a university repository Figure 2 depicts *Conceivable State Space* and *Lawful State Space* graphs for a Publication X. We would like to indicate that Publication is not the only business object in this example – also "Notification from Publisher" is a business object, CC licence, etc., but due to limited space we do not add analysis of other business objects. *Conceivable State Space* and *Lawful State Space* graphs were created using formalisms defined in Section 4:

1. LSS was created using formal definition LSS = {S, SL} – which represents a sequence of *Lawful States* and what are *Properties* of *Thing* for the lawful states:

LSS_{Px} = {{Registered, {Title X, Registered}}, {Open Access, {Title X, Registered, Yes}}, {CC Licence Chosen, {Title X, Registered, Yes, CC BY}}, {Full Text Version Chosen, {Title X, Registered, Yes, CC BY, Pre-Print, Yellow}}, {Publication Confirmed, Title X, Confirmed, Yes, CC BY, Pre-Print, Yellow}}.

2. CSS was created using formal definition CSS = {S, T} – which represents all possible sequences of states for Publication:

CSS$_P$ = {{Registered, Add Publication}, {Open Access, Choose OA Option}, {Not Open Access, Archive Internally}, {CC Licence Chosen, Choose CC Licence}, {Full Text Version Chosen, Choose Full Text Version}, {Publication Confirmed, Confirm OA Archiving}, {Publication Cancelled, Cancel OA Archiving}}.
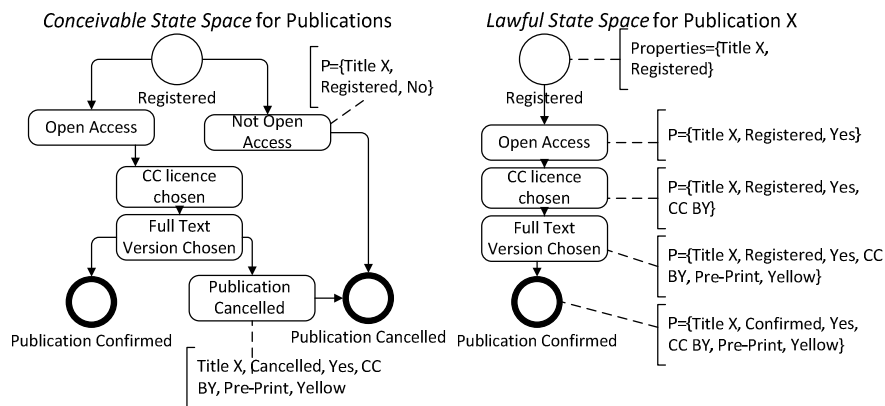

Fig. 2. Conceivable and lawful state spaces for a publication in electronic submission example.

6 Conclusions

Compliance between business process models and object state spaces are especially required in data-driven processes – in any process model that is based on data and manipulates with business objects. This paper presents an on-going research towards supporting compliance between business process models and lawful state space of business objects. BWW model is used as the foundation, since it allows straightforwardly addressing the lawful and conceivable state spaces of business objects. BPMN and ArchiMate modelling languages do not have elements that support explicit declaration of object states, including *State Law, Conceivable State Space, Lawful State Space, History, Conceivable Event Space,* and *Lawful Event Space.* The main contribution of this paper is a formalized solution for providing compliance between business process models and lawful states of business objects that has a capacity to support organizations in ensuring compliance between business process models and regulations.

With regards to tool support further research involves implementation of modelling environment capable of maintaining state spaces of business objects.

References

1. Weske, M.: Business Process Management. Springer Berlin Heidelberg, Berlin, Heidelberg (2012).
2. Goedertier, S., Vanthienen, J.: Designing compliant business processes with obligations and permissions. Bus. Process Manag. Work. 5–14 (2006).
3. Lohmann, N.: Compliance by design for artifact-centric business processes. Inf. Syst. 7241, (2013).
4. Müller, D., Reichert, M., Herbst, J.: A new paradigm for the enactment and dynamic adaptation of data-driven process structures. Adv. Inf. Syst. Eng. 48–63 (2008).
5. Meyer, A., Polyvyanyy, A., Weske, M.: Weak Conformance of Process Models with respect to Data Objects. ZEUS. 74–80 (2012).
6. OMG: Business Process Model and Notation 2.0, www.bpmn.org.
7. The Open Group: ArchiMate 2.0 Specification, http://goo.gl/7gC5B.
8. Chinosi, M., Trombetta, A.: BPMN: An introduction to the standard. Comput. Stand. Interfaces. 34, 124–134 (2012).
9. Silver, B.: BPMN Method and Style with Implementer's Guide. Cody-Cassidy Press (2011).
10. Softwareag: ARIS ArchiMate Modeler, http://goo.gl/WI76E.
11. Enterprise Architecture modeling software QPR EnterpriseArchitect, http://goo.gl/8Ots6h.
12. Rosemann, M., Recker, J., Indulska, M., Green, P.: A Study of the Evolution of the Representational Capabilities of Process Modeling Grammars. Adv. Inf. Syst. Eng. 447–461 (2006).
13. Penicina, L., Kirikova, M.: Towards Completeness and Lawfulness of Business Process Models. Perspect. Bus. Informatics Res. Lect. Notes Bus. Inf. Process. Vol.158, 63–77 (2013).
14. Penicina, L.: Linking BPMN, ArchiMate, and BWW: Perfect Match for Complete and Lawful Business Process Models? Short Pap. Proc. 6th IFIP WG 8.1 Work. Conf. Pract. Enterp. Model. (PoEM 2013). Vol-1023, 156–165 (2013).
15. Wand, Y., Weber, R.: On the ontological expressiveness of information systems analysis and design grammars. Inf. Syst. J. 3, 217–237 (1993).
16. Bunge, M.: Treatise on Basic Philosophy: Vol. 4: Ontology II: A World of Systems. (1979).
17. Recker, J., Indulska, M., Rosemann, M., Green, P.: Do Process Modelling Techniques Get Better ? A Comparative Ontological Analysis of BPMN. Campbell, Bruce, Underwood, Jim, Bunker, Deborah 16th Australas. Conf. Inf. Syst. (2005).
18. Ryndina, K., Küster, J., Gall, H.: Consistency of business process models and object life cycles. Model. Softw. Eng. 80–90 (2007).
19. Küster, J., Ryndina, K., Gall, H.: Generation of business process models for object life cycle compliance. Bus. Process Manag. 165–181 (2007).
20. Bons, R.W.H., Lee, R.M., Wagenaar, R.W., Wrigley, C.D.: Modelling inter-organizational trade using Documentary Petri Nets. Proceedings of the Twenty-Eighth Hawaii International Conference on System Sciences, vol.3. pp. 189–198. IEEE Comput. Soc. Press (1996).
21. OMG: UML 2.0 Superstructure specification, http://www.omg.org/spec/UML/2.0/Superstructure/PDF.
22. Governatori, G., Milosevic, Z., Sadiq, S.: Compliance checking between business processes and business contracts. 2006 10th IEEE International Enterprise Distributed Object Computing Conference (EDOC'06). pp. 221–232. IEEE (2006).
23. Aalst, W. Van der, Weske, M., Grünbauer, D.: Case handling: a new paradigm for business process support. Data Knowl. 1–36 (2005).