BPMN 2.0 Serialization - Standard Compliance Issues and Evaluation of Modeling Tools

Matthias Geiger, Guido Wirtz

Distributed Systems Group
University of Bamberg
An der Weberei 5
D-96047 Bamberg, Germany
{matthias.geiger, guido.wirtz}@uni-bamberg.de

Abstract: *Business Process Model and Notation* (BPMN) 2.0 process models are used more and more, both in practice as in academia. Although academic research mainly focuses on sophisticated semantic checks and extensions there still exist problems in the basic usage of BPMN. This paper investigates issues in BPMN model serializations which arise as a result of the complexity and inconsistency of the standard document. We present a set of serialization constraints as a starting point for sophisticated compliance checks on serialized BPMN models. Furthermore, these constraints are used to perform an evaluation of current modeling tools. This evaluation reveals that the creation of standard compliant models is still a non-trivial endeavor.

1 Introduction and Motivation

The *Business Process Model and Notation* (BPMN) [OMG11a] is an international standard for process modeling developed and maintained by the *Object Management Group* (OMG). It is widely accepted in practice and in academia, as demonstrated by various BPMN process modeling tools, engines for executing BPMN processes and also in various academic papers, see for instance [CT09, DDO08, LVD09, WG08], BPMN is a topic of research. The first versions of BPMN [BPM04, OMG08, OMG09] focus on the standardization of a set of graphical shapes for process modeling - this was also denoted by the former meaning of the acronym: *Business Process Modeling Notation*. For that reason, the main scope of BPMN was conceptual modeling of business processes for documentation and visualization purposes.

With version 2.0 [OMG11a] the standard has been enhanced by an (informal) definition of execution semantics to provide executability on compliant BPMN engines. By adding executability for process models, the complexity of the standard (and also of the resulting models) naturally is increased: Especially, aspects such as data handling which have not been covered in the graphical models of former versions have to be defined now.

A problematic aspect of the former BPMN versions is that no serialization format has been standardized. This led to a plethora of different serialization formats which hampers model

exchange between tools and engines: Modeling tools used their own proprietary file formats, but also mappings and transformations to other standards had been used. Especially, for processes which should be executable, a mapping to the *Web Services Business Process Execution Language* (WS-BPEL) [OAS07] was intended as the serialization format for executable service-based process models and has been considered and evaluated in various academic papers [RM06, KMWL09, WDGW08, ODtHvdA08]. A mapping to WS-BPEL is still part of the BPMN standard document [OMG11a, Sec.14], but executable BPMN models based on the official WS-BPEL-mapping have some portability issues [LW13].

However, it is not necessary any longer to use WS-BPEL to define executable processes as the implementation of official serialization formats is the second important extension in [OMG11a]. There are two XML-based formats proposed in [OMG11a, Sec. 15]: *XML Metadata Interchange* (XMI) [OMG11b] and a serialization based on *XML Schema Definitions* (XSDs) [W3C04b, W3C04a]. For both formats the OMG provides normative schema definitions[1]. Especially the XSD-based serialization is referred to throughout the whole standard document and is already used in various modeling tools and engines.

Both standardized serialization formats have a serious issue: They do not ensure the correctness of serialized BPMN models. In [OMG11a] various requirements and constraints are defined which have to be respected in well-formed, standard compliant models. These constraints range from graphical rules (such as the appearance of shapes) to rules for execution semantics. But there do also exist hundreds of constraints which are relevant for correct model serialization. Using the standardized serialization formats most of the structural constraints can be verified by performing checks such as a schema validation. But a lot of other rules cannot be checked in this way. This implies that just using the standardized formats does not guarantee that the model is BPMN compliant. The problem is exacerbated as [OMG11a] provides no list of all constraints which well-formed models have to respect. Instead, rules and constraints are spread all over the standard document in tables, figures, XML schema excerpts and the running text. As we sketched in previous work [GW13], besides vendor policy this is a main issues which hinders model exchange between different tools. To be able to determine whether a BPMN model is *correct* by means of standard compliance, it is important to know which rules models have to adhere.

The paper at hand provides two main contributions: First, we present an extensive list of constraints relevant for correct serialization stated in [OMG11a]. These rules are independent from the concrete underlying serialization mechanism. Therefore, they can be used to validate the standard compliance of models in the standardized as well as in proprietary formats. Furthermore, we assess which constraints are already covered by the XSD based serialization format and highlight some issues and inconsistencies. Second, we evaluate the serialization mechanism of several modeling tools. The evaluation consists of checks whether the standardized serialization format is supported and of an evaluation of the generated models' standard compliance. The focus of our analysis is the assessment of whether the tools implement checking mechanisms for the constraints we revealed.

The remainder of the paper is organized as follows: In the next section, the approach of extracting the constraints from [OMG11a] and the resulting set of BPMN serialization

[1] see: `http://www.omg.org/spec/BPMN/2.0/`

constraints is presented. The subsequent section evaluates how model serialization is implemented in state-of-the-art modeling tools. Related work is summarized and assessed in section 4. Finally, section 5 concludes the work and gives an outlook on future work.

2 BPMN Serialization Constraints

Before presenting our approach and results, we have to clarify the scope and limitations of our work: We strictly focus on the serialization of BPMN process models. We provide an overview of constraints stated in [OMG11a] regarding the serialization of BPMN process models. Although bound to serialization, we do not limit the standard screening to the normative XML-based serialization formats. Therefore, the set of rules is technology independent and can be used to analyze all types of BPMN serialization formats.

The following aspects are not covered in this work: Requirements regarding the visual appearance of BPMN shapes are out of scope. The serialization of BPMN diagrams using the *BPMN Diagram Interchange* format [OMG11a, Sec. 12] is left out as well. Moreover, we exclude all aspects regarding the execution of BPMN process models on compliant engines. First and foremost this involves all advanced execution semantics aspects but also all constraints affecting instance attributes and variables.

Due to the limitations of space, it is not possible to present all constraints here. A detailed description of our work and all extracted 611 rules can be found in a technical report [Gei13].

2.1 Constraint Categorization and Extraction Approach

The constraints stated in [OMG11a] can be divided and categorized in four major categories which are described hereafter.

Basic Attribute/Sub Element Cardinality (CARD): The most essential constraints define the general structure of BPMN models. The standard comprises a description of all possible elements, no matter whether they are depicted graphically or not, their attributes and their relations. In particular, for each attribute and model association, it is of interest whether the attribute is mandatory or whether some minimum or maximum occurrence constraints apply.

Basic Value Restrictions and Default Values (VAL): This addresses value restrictions, such as an enumeration of allowed values, and the definition of default values.

Basic Reference Constraints (REF): Another important aspect is the usage of references in [OMG11a]. BPMN allows to reuse and to refer to specific elements. The main source for references in concrete model instances is the definition of control and message flows. Each control flow link is realized as a *SequenceFlow* element which has to reference a source and a target element. The same applies to the definition of *MessageFlows*. In such cases an element can be referenced by other elements through a unique identifier and reference attributes. But reuse through referencing

is also possible for a lot of other elements. An example is the definition of a *Message* which is referenced by a *SendTask* and a corresponding *ReceiveTask* using their *messageRef* attribute [OMG11a, p.159-162].
Two aspects are important for each reference: First, the reference must be resolvable that is, the referenced item must exist in the same model or must be imported from another model or artifact. Second, in most cases only specific elements are allowed to be referenced. In the previous example, only *Messages* are valid elements. If the *messageRef* attribute references another element (e.g., an *Operation*), this would violate the model constraints in [OMG11a].

Extended Constraints (EXT): All other rules revealed in [OMG11a] are categorized as extended constraints. Typical rules for this category are constraints which apply only under certain preconditions. A frequently used precondition is that constraints only apply, if the process is defined as executable. Besides, restrictions which cannot be expressed by the basic cardinality, value and reference constraints, such as interdependence between model elements, are also listed as extended constraints.

The approach of determining rules basically consists of an in-depth analysis of the standard document. The identification and extraction of rules for the former three categories CARD, VAL and REF is rather straightforward: The standard document [OMG11a] provides tables listing all attributes and relations for (almost) each BPMN element. These tables provide (among other information) all aspects which are relevant to extract constraint definitions: attribute names, type definitions, value and cardinality restrictions.

More crucial is the definition of extended rules. Only some aspects may be derived from the previously mentioned tables. However, the majority of the extended constraints are spread over the standard's running text. In this cases not only the whole document is relevant, but also some interpretation is needed to identify requirements. Moreover, some other rules are not mentioned explicitly, nonetheless they are important. An example is that at least one *messageEventDefinition* must be present, if a *StartEvent* is target of a *MessageFlow* definition.

2.2 Results

In total we identified more than 600 different constraints. Figure 1 shows an overview of the distribution to the different categories. For a complete overview please refer to [Gei13]. However, the main findings and peculiarities are described in this section.

The standard [OMG11a] defines 311 different attributes and associations (CARD) for 108 BPMN elements. For each attribute the attribute name, the datatype and the required cardinality has been extracted. In combination with the defined inheritance structure these rules already allow the creation of structural correct BPMN models. It is problematic that the tables are not always well aligned with the UML class diagrams which are also included in [OMG11a]. Both, class diagrams and tables, contain information about the minimum and maximum occurrence of the relations between BPMN model elements and in some cases the information is inconsistent. For example, the *operationRef* attribute of

a *SendTask* is marked as optional in the class diagram (`[0..1]`) but the corresponding table states that the attribute is mandatory [OMG11a, p.160-161].

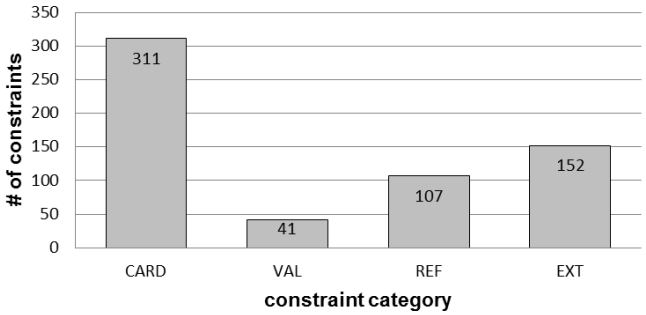

Figure 1: Number of extracted constraints by category

The 41 extracted value restrictions (VAL) can be divided in two groups: The first group covers default values for basic datatypes such as *boolean* and *string*. The second group comprises enumerations of allowed values. An example for this is the definition of allowed values for the attribute *gatewayDirection* for BPMN *Gateways*. This attribute indicates whether the gateway splits or merges the control flow and only the values *Unspecified*, *Converging*, *Diverging* and *Mixed* are valid [OMG11a, p.91].

As mentioned before, flow definitions (*Sequence* and *Message Flow*) are one important example for reference usage (REF). But the standard comprises 107 different associations between elements which are implemented as references. In order to determine the reference correctness for each reference definition the allowed datatypes have been determined.

Finally, 152 extended constraints (EXT) have been identified in the standard document. The complexity of the extended constraints varies. Examples for simple rules are constraints which only apply if the process is defined as *"executable"* that is, the attribute *isExecutable* of a process definition is set to `true`. But also more complex rules affecting various elements have been identified.

Generally, it is hard to prove that our list of constraints is complete and free of errors. Especially in cases where interpretation is needed or rules are stated only implicit, it is essential to ensure the quality and correctness of our extracted rules. However, we ensured the quality by internal discussions in our group and comparing our results with a less extensive set of BPMN constraints provided by Silver [Sil11]. And also the interpretations and according solutions implemented in different modeling tools have been considered. Moreover, we published all constraints and a technical report [Gei13] on our web page[2] and are looking forward for feedback from the community for an ongoing process of improvement for the constraint set.

[2] `http://www.uni-bamberg.de/pi/bpmn-constraints`

2.3 XSD-based Standard Serialization

As mentioned in section 1 the standard is closely linked to the normative XSDs and also more and more modeling tools support the import or export in the XSD-based serialization format. Therefore, it is of particular interest which standard constraints are already covered when using this format and whether a schema validation is able to reveal violations.

The structural aspects defined in the cardinality constraints are directly mappable to XML schema definitions. So, it is no surprise that most of these constraints are correctly implemented in the normative XSDs. Exceptions are some aspects which are not directly viable using XSDs, such as multiple inheritance, or deviations which depend on modeling decisions. An example is the implementation of mandatory attributes with default values. Using XML schema definitions, it is sufficient to define a default value and leave out the definition that an attribute is mandatory, as the default value always applies when no other value is defined (and therefore a value is always available). Another modeling decision is the omission of bi-directional references on XML level. An example is the attribute *boundaryEventRefs* for *Activities*. This attribute is omitted as each *BoundaryEventDefinition* references the activity to which it is attached by the attribute *attachedToRef*. Thus, it is still clearly defined which *BoundaryEventDefinition* is attached to which *Activity*. There also exist several clear violations of CARD constraints: The most frequent violation is that attributes are defined as mandatory in the standard document but the XSD marks them as optional. This is the case for 24 constraints. But, four other cardinality requirements are enforced incorrectly in the normative XSD.

Default values can be easily defined in schema definitions as well as value enumerations based on basic datatypes such as strings. But, four out of the 41 VAL constraints are implemented faultily, as required default values are not implemented. Two more inconsistencies exist affecting the attributes of the elements *Transaction* and *BoundaryEvent*. There, default values are defined in the XSD which are not stated in [OMG11a].

To implement references, BPMN proposes two different mechanisms [OMG11a, p.476-477]: If an element must be defined in the same file, the reference is realized using an *xs:IDREF* [W3C04b]. If references potentially cross file borders the reference attribute's datatype is set to *xs:QName* [W3C04b]. A schema validation can be used to determine whether an *xs:IDREF* is resolvable [W3C04a]. Missing element definitions for *xs:QName* references cannot be detected. Moreover, in both cases the reference type violations are undetectable using a basic schema validation and therefore there is no support for any of the 107 reference constraints (REF).

As with the extraction, the XSD-based implementation of extended constraints (EXT) is hard. The observance of most rules can not be enforced by schema definitions. Out of the 152 extended constraints only seven are directly implemented in XSDs. Examples are constraints regarding the mutual exclusion of attributes (implemented as sub-elements) using the *xs:choice* [W3C04a] operator.

To conclude, the XSD-based serialization format introduced in [OMG11a] is a good starting point to define standard compliant BPMN process models. Especially the basic model structure and value restrictions are well covered: About 91% of the CARD and 90% of

the VAL constraints are enforced by the XSDs. However, due to the missing reference checking support and only nominal (5%) support for the extended rules, the total constraint implementation is limited to about 54% of all 611 revealed constraints. Figure 2 shows the rule coverage in a graphical form.

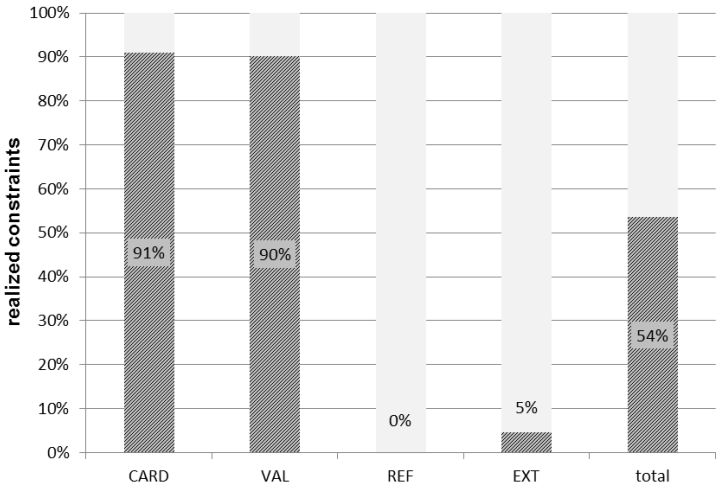

Figure 2: XSD-based serialization: constraint implementation coverage (percentage chart)

Regarding those numbers, it is evident that the usage of XML schema validation alone is not sufficient to enforce the correctness of models. Tool developers and end users who use such models need to perform more sophisticated checks to ensure that the model at hand is in fact standard compliant.

3 Evaluation of Existing Modeling Tools

Today, most BPMN modeling tools provide structural and semantic checks to some degree which enhance the abilities of XML schema validations. We compared and analyzed six BPMN editors regarding their ability to produce BPMN compliant models and their support to enforce the rules presented in section 2.

3.1 Analyzed tools

The analysis of BPMN editors comprises six different process modeling tools namely: BizAgi Process modeler[3], camunda Fox modeler[4], eclipse BPMN2 modeler[5], itp-commerce Process Modeler 6 for Visio[6], Signavio Process Modeler[7] and Yaoqiang BPMN Editor[8]. All tools are dedicated for BPMN process modeling. Several tools are freely available as freeware or open source software, but we also evaluate two commercial editors (Signavio and itp-commerce).

Table 1 shows an overview of the tested modeling tools. Besides the vendor and the editor name, the actual version under test is stated. Furthermore, it is indicated whether the tool is commercial (*comm.*), freely available (*freeware*) or published as open source (*OS*). The base platform is depicted in the subsequent row. Some editors are implemented as a standalone tool, three editors are implemented as plugins for eclipse or Microsoft Visio and one editor is cloud-based and usable through a standard web browser. Moreover, for each modeling tool the default serialization format and the ability to import or export in the standardized XSD-based format is expressed.

Vendor/Developer	BizAgi	camunda	eclipse	itp-commerce	Shi Yaoqiang	Signavio
Tool name	Process modeler	Fox modeler	BPMN2 modeler	Process Modeler 6 for Visio	Yaoqiang BPMN editor	Process Modeler
Version	2.4.0.8	2.0.12	0.2.6	6.3438	2.1.36	7.0.0
License	freeware	OS	OS	comm.	OS	comm.
Platform	standalone	eclipse	eclipse	MS Visio	standalone	cloud-based
Default Serialization Format	*prop.*	BPMN XSD	BPMN XSD	MS Visio	BPMN XSD	*unknown*
Import/Export support (XSD-based serialization)	$-/-$	$+/+$	$+/+$	$(+)/+$	$+/+$	$+/+$

Table 1: Overview of evaluated BPMN process modeling tools

3.2 Evaluation Method

To assess the six different modeling tools we perform a three-step approach: First, we evaluate the supported BPMN conformance level and validation features of the tools under test. Depending on the determined supported conformance class, we check the modeling

[3] http://www.bizagi.com/download/
[4] http://www.camunda.org/design/modeler.html
[5] http://www.eclipse.org/bpmn2-modeler/
[6] http://www.itp-commerce.com/
[7] http://www.signavio.com
[8] http://sourceforge.net/projects/bpmn/

abilities and the serialization correctness in an second step. Finally, some advanced features and extended constraints are checked. Each step is described in more detail in the following paragraphs.

Supported conformance level and validation features: [OMG11a, Sec.2] introduces several *conformance levels* which define different levels of BPMN support. Especially the conformance level *Process Modeling Conformance* is relevant for the evaluation at hand. This level is refined to four sub-classes: *Descriptive*, *Analytic*, *Common Executable* and *Full Process Modeling Conformance* [OMG11a, p.2]. Depending on the level a modeling tool must support a specific set of BPMN elements and attributes [OMG11a, p.2-7]. In order to assess the modeling abilities of the tested modelers it is important to know which conformance class is supported by each tool. As the tools do not indicate which level they support, the level has to be determined by analyzing the palette of supported elements and their attributes. Furthermore, for each editor it is checked whether validation features are implemented. If validation techniques are used, we check whether the validation has to be performed manually, whether it is started automatically when the model is saved or exported, or whether the editor supports real-time validation during modeling.

Modeling abilities and serialization correctness: Depending on the supported conformance classes the editor specific modeling abilities are assessed in the next step. To achieve this, for each conformance level example processes are modeled with each editor. The resulting models are checked, whether a) it was possible to recreate the model, b) the model passes an XML schema validation and c) the model violates some other constraints.

The example models for the descriptive and analytic sub-classes are gathered from the *BPMN 2.0 by example* document provided by the OMG [OMG10]: We evaluate the ability of the editors to recreate the "Small Examples introducing Core Concepts" [OMG10, Sec.5]. In order to evaluate the modeling abilities for the common executable and full conformance sub-classes we use process models created in a student project containing advanced features regarding Web Services usage and data handling.

Evaluation of advanced features and extended constraint validation: We select several well known correctness aspects from our list of extended constraints[9] and check whether they are correctly implemented in the modeling tools.

- In [OMG11a, p.54] it is stated that *"[i]mporting Xml Schema 1.0, WSDL 2.0 and BPMN 2.0 types MUST be supported"*. We assess whether the tools support this requirement.
- BPMN allows the reuse of elements and processes defined in other files. So, it is evaluated whether tools are able to reference elements across file-borders.
- EXT.028/031: *SequenceFlows* must not cross the borders of pools; *MessageFlows* must cross pool borders.
- EXT.096/104: For *StartEvents* no incoming *SequenceFlow*, for *EndEvents* no outgoing *SequenceFlow* is allowed.

[9] See [Gei13] for a detailed description of the constraints.

- EXT.100: For sub-processes only “none” *StartEvents* are allowed.
- EXT.016-019: The attribute *gatewayDirection* of *Gateways* defines whether the gateway is splitting (attribute value “diverging”) or joining (“converging”) the control flow. If a gateway has multiple incoming and outgoing sequence flows, the attribute must have the value “mixed” or “unspecified” (attribute mandatory for common executable and full conformance only)
- EXT.057: An *Event Sub-Process* must not have any incoming or outgoing *SequenceFlows*.

3.3 Evaluation Results

In the following, the main findings are described and Table 2 gives an overview of our evaluation results. In the table “$+$” denotes that the feature/constraint under test is fully supported, “$-$” is used if a feature is not supported and partial support is indicated by “$(+)$”.

Regarding the conformance classes all tools support at least the analytic sub-class. However, some elements and minor aspects are missing for most tools. So technically the editors do not fully support the conformance classes. For example, the BizAgi editor is not able to model *CallActivities* which are mandatory for the descriptive conformance class. Generally the BizAgi Process modeler is a particular case. It is not able to serialize the model in the standardized XML format. Thus, it is hard to determine the supported conformance class and it is not possible to perform schema validation of the serialized models.

Some kind of model validation is implemented in each modeling tool, but only the itp-commerce and the Yaoqiang editor support real-time checks during modeling. In all other tools the validation has to be started manually or the validation is only performed when saving or opening a file.

It is mostly possible to recreate the example models from [OMG10] in the different tools, but some minor issues exist: Eclipse and Yaoqiang are not able to bind a *TextAssociation* to *SequenceFlows* which is allowed by [OMG11a], resulting in only partial support ($(+)$) for the descriptive sub-class. The BizAgi editor fails to provide support for modeling *Message* elements as required by the analytic conformance sub-class. Modeling processes for the common executable and full conformance class was only performed for eclipse and Yaoqiang and both editors provide good support for this conformance classes, but eclipse is not able to model event sub-processes.

Import and usage of WSDL, XSD and BPMN files in a BPMN model is supported by eclipse and the Yaoqiang editor. Itp-commerce provides support for BPMN imports only. Thus, only these three editors can handle cross-file references for sub-processes and other BPMN elements. It is to note, that Signavio supports referencing in the cloud-based application, but when the model is serialized the referenced elements are either left out or included in the model, instead of referencing the called element.

Vendor/Developer	BizAgi	camunda	eclipse	itp-commerce	Shi Yaoqiang	Signavio
Tool name	Process modeler	Fox modeler	BPMN2 modeler	Process Modeler 6 for Visio	Yaoqiang BPMN editor	Process Modeler
Supported conformance level and validation features						
Supported conformance class	*(analytic)*	*(analytic)*	*(full)*	analytic	*(full)*	analytic
Model validation	manual	while opening model	opening/ saving	real-time/ manual	real-time	manual/ saving
Modeling abilites and serialization correctness						
Descriptive						
creation	+	+	(+)	+	(+)	+
schema valid	*n/a*	+	+	+	+	+
Analytic						
creation	(+)	+	+	+	+	+
schema valid	*n/a*	+	+	+	+	+
C. Executable						
creation	*n/a*	*n/a*	+	*n/a*	+	*n/a*
schema valid	*n/a*	*n/a*	+	*n/a*	+	*n/a*
Full						
creation	*n/a*	*n/a*	(+)	*n/a*	+	*n/a*
schema valid	*n/a*	*n/a*	+	*n/a*	+	*n/a*
Evaluation of advanced features and extended constraint validation						
WSDL/XSD/ BPMN import	−	−	+	(+), *only BPMN*	+	−
Cross-file references	−	−	+	+	+	−
EXT.028/031	+	+	+	+	+	+
EXT.096/104	+	−	+	+	+	+
EXT.100	−	−	−	+	+	−
EXT.016-019	*n/a*	*n/a*	+	+	(+)	(+)
EXT.057	+	−	*n/a*	+	+	+

Table 2: Overview: Evaluation results

The constraints EXT.028 and EXT.031 stating sequence and message flow connection rules are well implemented in all of the six tools. Either, the editors do not allow invalid connections or invalid connections are marked during a validation. The *SequenceFlow* restrictions for start and end events are also covered in all tools, except for the camunda modeler which does not check the adherence to this rules. Rule EXT.100 regarding allowed *StartEvent* type for sub-process is checked by itp-commerce and Yaoqiang. The *gatewayDirection* constraints (EXT.016-019) cannot be assessed for BizAgi and camunda as the attribute is not existent in the models and their serialization. Yaoqiang and Signavio cannot handle "mixed" gateways, but eclipse and itp-commerce implement all rules correctly. EXT.057 is implemented correctly in most tools. Camunda does not implement the constraint check and eclipse is not able to model event sub-processes.

All in all, the itp-commerce editor has implemented most of the rules checked in our evaluation. A drawback is that the editor is limited regarding modeling executable processes and does not support the usage of WSDL and XSD imports.

4 Related Work

Previous research on the correctness of BPMN mainly focuses on establishing formal semantics for BPMN and providing support for semantic validation, verification and correctness checks for BPMN models (see for instance [DDO08, LVD09, WG08, GD12] without claiming to be complete). We also evaluate the "correctness" of BPMN models, but our work targets another level of abstraction: Correctness in this work refers to standard compliance, especially regarding model serialization.

Nowadays, various extensions for specific domains, such as cloud application management [KBBL12], and aspects such as security [BHLR12] and social BPM [BFV12] are developed. In contrast to this, our work is limited to the standard [OMG11a], domain-independent and (regarding the set of constraints) technology-independent.

An approach related closely to BPMN serialization is [CT09]. The authors develop a simplified meta model and an XML-based serialization format for BPMN 1.1 [OMG08]. For this serialization format they provide a mechanism to check references and also complex constraints can be checked using XPath expressions. Unfortunately, the approach is not applicable for BPMN 2.0 process models, as the proposed serialization format is not standard compatible any more.

Besides academic work, a set of BPMN constraints, which comprises 39 rules, is provided by Silver in [Sil11]. Our constraint set is far more extensive for two reasons: First, we do not exclude all cardinality, value and reference constraints which are already covered by the normative XSDs. Second, we do not limit our approach to the *"Analytic Process Modeling Conformance subclass"* as Silver does. And recently (since January 2013) also the OMG recognized model serialization and interchange between tools as an important issue. They initiated the *BPMN Model Interchange Working Group* (BPMN MIWG)[10] with the goal to provide support for modeling tool developers and to identify issues in the standard which inhibit interchange of models between tools. Planned outputs are test cases, feature tests, a set of BPMN 2.0 issues and interchange guidelines[11]. The group is still working in an initial project phase and no official outcomes have been released yet.

5 Conclusion and Outlook

In our work we have presented an extensive and technology-independent set of BPMN serialization constraints. This list is a basis for tool developers as well as for end users who want to check the standard compliance of serialized BPMN models. All in all, we have identified 611 different rules in [OMG11a] and have categorized them in four different categories. An analysis of the standardized XSD-based serialization format of BPMN shows that only 54% of all rules are enforcable by schema validation. Especially, extended constraints cannot be checked by such a validation. Therefore, the correctness of models and their serialization depends on the ability of the modeling tools to enforce the observance

[10] `http://www.omgwiki.org/bpmn-miwg/doku.php`

[11] see: `http://www.omgwiki.org/bpmn-miwg/doku.php#outputs`

off all constraints.

Our evaluation shows that state-of-the-art modeling editors already implement the revealed constraints partly. The evaluated tools are able to generate and serialize standard compliant models with a basic feature set. However, the internal validation mechanisms of the tool versions tested (see Table 1) still do not cover all revealed constraints which have to be respected in correct BPMN models. Moreover, the evaluation also shows that the serialization of models is an important first step, but also the diagram serialization is still an issue. Thus, we plan to broaden the scope of our rule set to cover BPMN Diagram Interchange (BPMN DI) constraints as well.

Due to the extensive set of rules, a manual review of the standard compliance of concrete models is not feasible. In order to provide tool independent conformance checks of BPMN models we plan to implement a validation suite which checks all serialization rules automatically. We have implemented automatic checks of all reference rules and are now extending the validation tool to check extended constraints.

References

[BFV12] Marco Brambilla, Piero Fraternali, and Carmen Vaca. BPMN and Design Patterns for Engineering Social BPM Solutions. In *Business Process Management Workshops*, volume 99 of *Lecture Notes in Business Information Processing*, pages 219–230. Springer Berlin Heidelberg, 2012.

[BHLR12] Achim D. Brucker, Isabelle Hang, Gero Lückemeyer, and Raj Ruparel. SecureBPMN: modeling and enforcing access control requirements in business processes. In *Proceedings of the 17th ACM symposium on Access Control Models and Technologies*, SACMAT '12, pages 123–126. ACM, 2012.

[BPM04] BPMI (Business Process Management Initiative). *Business Process Modeling Notation (BPMN) Version 1.0*, May 2004.

[CT09] Michele Chinosi and Alberto Trombetta. Modeling and validating BPMN diagrams. In *Commerce and Enterprise Computing, 2009. CEC'09. IEEE Conference on*, pages 353–360. IEEE, 2009.

[DDO08] Remco M. Dijkman, Marlon Dumas, and Chun Ouyang. Semantics and analysis of business process models in BPMN. *Information and Software Technology*, 50(12):1281 – 1294, 2008.

[GD12] Pieter Van Gorp and Remco M. Dijkman. A visual token-based formalization of BPMN 2.0 based on in-place transformations. *Information and Software Technology*, 2012.

[Gei13] Matthias Geiger. BPMN 2.0 Process Model Serialization Constraints. Bamberger Beiträge zur Wirtschaftsinformatik und Angewandten Informatik, no. 92, Otto-Friedrich-Universität Bamberg, May 2013.

[GW13] Matthias Geiger and Guido Wirtz. Detecting Interoperability and Correctness Issues in BPMN 2.0 Process Models. In *Proceedings of the 5th Central-European Workshop on Services and their Composition (ZEUS), Rostock, Germany*, CEUR Workshop Proceedings, pages 39–42. CEUR-WS.org, Feb 2013.

[KBBL12] Oliver Kopp, Tobias Binz, Uwe Breitenbücher, and Frank Leymann. BPMN4TOSCA: A Domain-Specific Language to Model Management Plans for Composite Applications. In *Business Process Model and Notation*, volume 125 of *Lecture Notes in Business Information Processing*, pages 38–52. Springer Berlin Heidelberg, 2012.

[KMWL09] Oliver Kopp, Daniel Martin, Daniel Wutke, and Frank Leymann. The Difference Between Graph-Based and Block-Structured Business Process Modelling Languages. *Enterprise Modelling and Information Systems Architectures*, 4(1):3–13, 2009.

[LVD09] Niels Lohmann, Eric Verbeek, and Remco Dijkman. PetriNet Transformations for BusinessProcesses ASurvey. In *Transactions on Petri Nets and Other Models of Concurrency II*, volume 5460 of *Lecture Notes in Computer Science*, pages 46–63. Springer Berlin Heidelberg, 2009.

[LW13] Jörg Lenhard and Guido Wirtz. Detecting Portability Issues in Model-Driven BPEL Mappings. In *Proceedings of the 25th International Conference on Software Engineering and Knowledge Engineering (SEKE'2013)*, Boston, USA, June 2013.

[OAS07] OASIS. *Web Services Business Process Execution Language*, April 2007. v2.0.

[ODtHvdA08] Chun Ouyang, Marlon Dumas, Arthur H.M. ter Hofstede, and Wil M.P. van der Aalst. Pattern-based translation of BPMN process models to BPEL web services. *International Journal of Web Services Research (IJWSR)*, 5(1):42–62, 2008.

[OMG08] OMG (Object Management Group). *Business Process Modeling Notation (BPMN) Version 1.1*, January 2008.

[OMG09] OMG (Object Management Group). *Business Process Modeling Notation (BPMN) Version 1.2*, January 2009.

[OMG10] OMG (Object Management Group). *BPMN 2.0 by Example*, June 2010.

[OMG11a] OMG (Object Management Group). *Business Process Model and Notation (BPMN) Version 2.0*, January 2011.

[OMG11b] OMG (Object Management Group). *OMG MOF 2 XMI Mapping Specification Version 2.4.1*, August 2011.

[RM06] Jan C. Recker and Jan Mendling. On the translation between BPMN and BPEL: Conceptual mismatch between process modeling languages. In *18th International Conference on Advanced Information Systems Engineering. Proceedings of Workshops and Doctoral Consortium*, pages 521–532. Namur University Press, 2006.

[Sil11] Bruce Silver. *BPMN method and style*. Cody-Cassidy Press, Aptos, CA, USA, 2nd edition, 2011.

[W3C04a] W3C. *XML Schema Part 1: Structures Second Edition*, October 2004.

[W3C04b] W3C. *XML Schema Part 2: Datatypes Second Edition*, October 2004.

[WDGW08] Matthias Weidlich, Gero Decker, Alexander Grosskopf, and Mathias Weske. BPEL to BPMN: The Myth of a Straight-Forward Mapping. In *On the Move to Meaningful Internet Systems: OTM 2008*, volume 5331 of *Lecture Notes in Computer Science*, pages 265–282. Springer Berlin Heidelberg, 2008.

[WG08] PeterY.H. Wong and Jeremy Gibbons. A Process Semantics for BPMN. In *Formal Methods and Software Engineering*, volume 5256 of *Lecture Notes in Computer Science*, pages 355–374. Springer Berlin Heidelberg, 2008.