IEEE Standard for eXtensible Event Stream (XES) for Achieving Interoperability in Event Logs and Event Streams

IEEE Computational Intelligence Society

Sponsored by the
Standards Committee

IEEE
3 Park Avenue
New York, NY 10016-5997
USA

IEEE Std 1849™-2016

IEEE Standard for eXtensible Event Stream (XES) for Achieving Interoperability in Event Logs and Event Streams

Sponsor

Standards Committee
of the
IEEE Computational Intelligence Society

Approved 22 September 2016

IEEE-SA Standards Board

Abstract: A grammar for a tag-based language whose aim is to provide designers of information systems with a unified and extensible methodology for capturing systems' behaviors by means of event logs and event streams is defined in the XES standard. An "XML Schema" describing the structure of an XES event log/stream and a "XML Schema" describing the structure of an extension of such a log/stream are included in this standard. Moreover, a basic collection of so-called "XES extension" prototypes that provide semantics to certain attributes as recorded in the event log/stream is included in this standard.

Keywords: event log, event stream, extensions, IEEE 1849™, system behavior, XML

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

PDF: ISBN 978-1-5044-2421-9 STD21181
Print: ISBN 978-1-5044-2422-6 STDPD21181

Translations

The IEEE consensus development process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE should be considered the approved IEEE standard.

Official statements

A statement, written or oral, that is not processed in accordance with the IEEE-SA Standards Board Operations Manual shall not be considered or inferred to be the official position of IEEE or any of its committees and shall not be considered to be, or be relied upon as, a formal position of IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position of IEEE.

Comments on standards

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE. However, IEEE does not provide consulting information or advice pertaining to IEEE Standards documents. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important that any responses to comments and questions also receive the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to comments or questions except in those cases where the matter has previously been addressed. For the same reason, IEEE does not respond to interpretation requests. Any person who would like to participate in revisions to an IEEE standard is welcome to join the relevant IEEE working group.

Comments on standards should be submitted to the following address:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
Piscataway, NJ 08854 USA

Laws and regulations

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document does not imply compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

Copyrights

IEEE draft and approved standards are copyrighted by IEEE under US and international copyright laws. They are made available by IEEE and are adopted for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making these documents available for use and adoption by public authorities and private users, IEEE does not waive any rights in copyright to the documents.

Participants

At the time this IEEE standard was completed, the XES Working Group had the following membership:

Wil van der Aalst, *Chair*
Christian Günther, *Vice Chair*

JC Bose
Josep Carmona
Marlon Dumas
Frank van Geffen
Sukriti Goel
Antonella Guzzo
Rania Khalaf
Rudolf Kuhn
Teemu Lehto
Felix Mannhardt
Marco Montali
Michael zur Muehlen
Zbigniew Paszkiewicz
Hajo Reijers
Alexander Rinke
Michal Rosik
Anne Rozinat
Pnina Soffer
Minseok Song
Keith Swenson
Walter Vanherle
Eric Verbeek
Lijie Wen
Moe Wynn

The following members of the individual balloting committee voted on this standard. Balloters may have voted for approval, disapproval, or abstention.

Giovanni Acampora
Michael Bauer
JC Bose
Keith Chow
Randall Groves
Christian Günther
Antonella Guzzo
Werner Hölzl
Noriyuki Ikeuchi
Piotr Karocki
Akhil Kumar
Edward McCall
Björn Molitor
Takahide Nogayama
Zbigniew Paszkiewicz
Michal Rosik
Christopher Turner
Wil van der Aalst
Eric Verbeek
Lijie Wen
Oren Yuen

Introduction

This introduction is not part of IEEE Std 1849-2016, IEEE Standard for eXtensible Event Stream (XES) for Achieving Interoperability in Event Logs and Event Streams.

Event logs contain information on how processes have evolved in running systems. As more and more systems capture such information, there is a need to be able to transfer this information from these running systems to a site where the information can be analyzed, either automatically by software from the computational intelligence field, or manually (at least in part) using such software.

This standard addresses this need by defining an eXtensible Event Stream (XES) structure for such event logs.

Furthermore, this standard defines the World Wide Web Consortium (W3C) Extensible Markup Language (XML) structure and constraints on the contents of XML 1.1 documents that can be used to represent XES instances, and a likewise structure (called XESEXT) that can be used to represent so-called extensions to this structure.

The purpose of this standard is to allow the generation and analysis of event logs using XML. This standard uses the W3C XML Schema definition language as the encoding, which allows for interoperability and the exchange of XES XML instances between various systems.

Contents

IEEE Standard for eXtensible Event Stream (XES) for Achieving Interoperability in Event Logs and Event Streams

1. Overview

1.1 Scope

This standard defines World Wide Web Consortium (W3C) Extensible Markup Language (XML) structure and constraints on the contents of XML 1.1 documents that can be used to represent extensible event stream (XES) instances.[1] A XES instance corresponds to a file-based event log or a formatted event stream that can be used to transfer event-driven data in a unified and extensible manner from a first site to a second site. Typically, the first site will be the site generating this event-driven data (for example, workflow systems, case handling systems, procurement systems, devices like wafer steppers and X-ray machines, and hospitals) while the second site will be the site analyzing this data (for example, by data scientists and/or advanced software systems).

To transfer event-driven data in a unified manner, this standard includes a W3C XML Schema describing the structure of a XES instance. To transfer this data in an extensible manner, this standard also includes a W3C XML Schema describing the structure of an extension to such a XES instance. Basically, such an extension provides semantics to the structure as prescribed by the XES instance. Finally, this standard includes a basic collection of such extensions.

1.2 Purpose

The purpose of this standard is to provide a generally acknowledged XML format for the interchange of event data between information systems in many applications domains on the one hand and analysis tools for such data on the other hand. As such, this standard aims to fix the syntax and the semantics of the event data which, for example, is being transferred from the site generating this data to the site analyzing this data. As a result of this standard, if the event data is transferred using the syntax as described by this standard, its semantics will be well understood and clear at both sites.

[1]Although XES is now an IEEE standard, XES was already used before it became an IEEE standard. Annex A shows the history of this "old" XES leading up to this IEEE standard containing the "new" XES, Annex B shows the main differences between the "old" XES and the "new" XES, whereas Annex C contains a list of tools that support the "old" XES, a list of data sets already published using the "old" XES, and a list of publications referring to the "old" XES.

2. Normative references

The following referenced documents are indispensable for the application of this document (i.e., they must be understood and used, so each referenced document is cited in text and its relationship to this document is explained). For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

ISO 639-1, Codes for the Representation of Names of Languages—Part 1: Alpha-2 Code.[2]

ISO 639-2, Codes for the Representation of Names of Languages—Part 2: Alpha-3 Code.

ISO 4217:2008, Codes for the Representation of Currencies and Funds.

ISO 8601, Data elements and interchange formats—Information interchange—Representation of dates and times.

XML Schema Part 1: Structures, Second Edition, W3C Recommendation (28 October 2004).[3]

XML Schema Part 2: Datatypes, Second Edition, W3C Recommendation (28 October 2004).[4]

3. Definitions, abbreviations, and acronyms

3.1 Definitions

For the purposes of this document, the following terms and definitions apply. The *IEEE Standards Dictionary Online* should be consulted for terms not defined in this clause. [5]

component: An extensible event stream (XES) element that may contain XES attributes, that is, a log, a trace, an event, or an attribute.

3.2 Abbreviations and acronyms

BPAF	Business Process Analytics Format
UUID	universally unique identifier
URI	uniform resource indentifier
XES	extensible event stream
XESEXT	XES extension
XML	Extensible Markup Language

[2]ISO publications are available from the ISO Central Secretariat, http://www.iso.org/. ISO publications are also available in the United States from the American National Standards Institute, http://www.ansi.org/.
[3]Available at: https://www.w3.org/TR/2004/REC-xmlschema-1-20041028/
[4]https://www.w3.org/TR/xmlschema-2/
[5]*IEEE Standards Dictionary Online* subscription is available at: http://dictionary.ieee.org.

4. XES metadata structure

4.1 Hierarchical components

4.1.1 Log component

A *log component* represents information that is related to a specific process. Examples of processes include handling insurance claims, using a complex X-ray machine, and browsing a website. A log shall contain a (possibly empty) collection of traces followed by a (possibly empty) list of events. The order of the events in this list shall be important, as it signifies the order in which the events have been observed.

If the log contains only events and no traces, then the log is also called a *stream*.

4.1.2 Trace component

A *trace component* represents the execution of a single case, that is, of a single execution (or enactment) of the specific process. A trace shall contain a (possibly empty) list of events that are related to a single case. The order of the events in this list shall be important, as it signifies the order in which the events have been observed.

4.1.3 Event component

An *event component* represents an atomic granule of activity that has been observed. If the event occurs in some trace, then it is clear to which case the event belongs. If the event does not occur in some trace, that is, if it occurs in the log, then we need ways to relate events to cases. For this, we will use the combination of a trace classifier and an event classifier, see 4.4.

4.2 Attribute component

Information on any component (log, trace, or event) is stored in *attribute components*. Attributes describe the enclosing component, which may contain an arbitrary number of attributes. However, no two attributes of the same component may share the same key; that is, every key shall occur only once in a single component.

For providing maximum flexibility, this standard allows for *nested* attributes, that is, attributes that themselves have child attributes. While this feature is necessary for efficient encoding of certain information types, it is *optional for tools to implement nested attributes*; that is, the feature is not strictly required in order to be compliant to this standard. Nevertheless, a tool that does not support nested attributes shall be able to read documents that feature nested attributes. These tools shall transparently ignore and discard any nested attributes, and, where feasible, alert the user to the fact that some information may not be available.

An attribute shall be either *elementary* or *composite*.

4.2.1 Elementary attributes

An *elementary attribute* is an attribute that shall contain an elementary (single, basic) value. In this standard, an elementary attribute shall be a *string attribute*, a *date and time attribute*, an *integer number attribute*, a *real number attribute*, a *Boolean attribute*, or an *ID attribute*.

4.2.1.1 String attributes

Valid values for a string attribute are values that conform to the *xs:string* datatype.

4.2.1.2 Date and time attributes

Values for a date and time attribute should be specified in UTC time (see ISO 8601[6]), also known as Zulu time, and represented as *xs:dateTime* datatype.

4.2.1.3 Integer number attributes

Valid values for an integer number attribute are values that conform to the *xs:long* datatype.

4.2.1.4 Real number attributes

Valid values for a real number attribute are values that conform to the *xs:double* datatype.

4.2.1.5 Boolean attributes

Valid values for a Boolean attribute are values that conform to the *xs:boolean* datatype.

4.2.1.6 ID attributes

Valid values for an ID attribute are values that conform to the ID datatype, that is, all string representations of universally unique identifiers (UUIDs).

4.2.2 Composite attributes

A *composite attribute* is an attribute that may contain multiple values. In this standard, a composite attribute shall be a *list attribute*.

4.2.2.1 List attribute

Valid values for the list datatype are all lists (series) of attribute values. The order between the child attributes in this list shall be important. In contrast to attributes enclosed in a component, attributes enclosed in such an attribute value list may share the same key. Note that the attribute value list is not the same as the list attribute: The former is contained in the latter, and the latter is a component but the former is not.

4.3 Global attributes

A log shall hold a (possibly empty) list of global attribute declarations. A global attribute declaration shall have a valid *key*, a valid *datatype*, and a valid *value* for the datatype. A global attribute declaration shall either be a global event attribute or a global trace attribute.

Global attributes are a required feature for compliance to this standard. Nevertheless, a defensive approach is recommended with respect to global attributes, as there is no way to undo a global declaration.

4.3.1 Global event attributes

Global event attributes are event attributes that are understood to be *available* and *properly defined* for each event in the log (be it in a trace or not). As a result, every event in the log shall contain an attribute with the given key and the given datatype, but possibly with a different valid value. The value provided for a global event attribute declaration is *only significant* in case an event *needs to be created* (for some reason) for which *no value is provided* for that attribute. In that case, the value of the declaration shall be used as the value for the attribute. In all other cases, the value of the declaration is insignificant and shall not be used.

[6]Information on references can be found in Clause 2.

4.3.2 Global trace attributes

Global trace attributes are trace attributes that are understood to be *available* and *properly defined* for each trace in the log. As a result, every trace shall contain an attribute with the given key and the given datatype, but possibly with a different valid value. The value provided for a global attribute declaration is *only significant* in case a trace *needs to be created* (for some reason) for which *no value is provided* for that attribute. In that case, the value of the declaration shall be used as the value for the attribute. In all other cases, the value of the declaration is insignificant, and shall not be used.

4.4 Classifiers

In this standard, there are no predefined attributes with any well-understood meaning per se. Instead, a log shall hold a (possibly empty) list of classifiers. *These classifiers are a mandatory feature of this standard*.

A classifier assigns an *identity* to each event that makes it comparable to others (via their assigned identity). Examples of such identities include the descriptive name of the event, the descriptive name of the case the event relates to, the descriptive name of the cause of the event, and the descriptive name of the case related to the event.

A classifier shall either be an *event classifier* or a *trace classifier*.

In case the log contains events that do not occur in a trace, it is necessary to be able to relate these events to cases. For this reason, we assume that one of the existing event classifiers provides the descriptive name of its case. Two events for which this classifier results in the same identity belong to the same case. Furthermore, we assume that one of the trace classifiers provides the descriptive name for the case. If this classifier and the event classifier mentioned earlier return the same identity, the corresponding events belong to the same case as the corresponding trace. As such, the event shall be appended to this trace in the same order as they appear in the log. If no matching trace exists, a new trace shall be constructed from these events again in the same order as they appear in the log.

4.4.1 Event classifiers

An event classifier shall be defined via an ordered list of attribute keys. The identity of the event shall be derived from the actual values of the attributes with these keys. An attribute whose key appears in an event classifier list shall be declared as a global event attribute before the event classifier is defined, as the actual value for the attribute is required by the event classifier.

4.4.2 Trace classifiers

A trace classifier shall be defined via an ordered list of attribute keys. The identity of the trace shall be derived from the actual values of the attributes with these keys. An attribute whose key appears in a trace classifier list shall be declared as a global trace attribute before the trace classifier is defined, as the actual value for the attribute is required by the trace classifier.

4.4.3 Event ordering

Within the context of a single trace, the ordering of events shall be important: An event that occurs in a log (be it in a trace or in the log itself) before another event that is related to the same trace shall be assumed to have occurred before that other event. However, the notion of a trace also depends on the trace and event classifiers selected by the user. As such, the notion of a trace is not necessarily predefined for a log. As a result, the notion of the event ordering may be affected by the choice of classifiers.

Whatever classifier is selected by the user, the ordering in the log shall be maintained: The first event that belongs to some trace shall be the first event encountered in the log (be it in a trace or in the log itself). For example, consider the event log that contains the following:

a) a trace containing events e_{11} and e_{12}

b) a trace containing an event e_{21}

c) a trace containing events e_{31}, e_{32}, and e_{33}

d) an event e_4

Now, assume that the user has selected an event classifier and a trace classifier that causes the events e_{11}, e_{31}, e_{33}, and e_4 to belong to the same trace. Then the ordering in this (classified) trace shall be e_{11}, e_{31}, e_{33}, and e_4.

4.5 Extensions

This standard does not define a specific set of attributes per component. As such, the semantics of the data attributes these elements do contain may necessarily be ambiguous, hampering the interpretation of that data.

This ambiguity is resolved by the concept of *extensions* in this standard. An extension defines a (possibly empty) set of attributes for every type of component. The extension provides points of reference for interpreting these attributes, and, thus, their components. Extensions, therefore, are *primarily a vehicle for attaching semantics to a set of defined attributes per component*.

Extensions have many possible uses. One important use is to introduce a set of commonly understood attributes that are vital for a specific perspective or dimension of event log analysis (and which may even not have been foreseen at the time of developing this standard). See Clause 7 for the current set of standard extensions.

Other uses include the definition of generally-understood attributes for a specific application domain (for example, medical attributes for hospital processes), or for supporting special features or requirements of a specific application.

An extension shall have a descriptive *name*, a *prefix*, and a *uniform resource identifier (URI)*. The prefix is the prefix of all attributes defined by the extension. This means the keys of all attributes defined by the extension shall be prepended with this prefix and colon separation character (like a namespace in XML). The URI is a unique URI that points to the definition of the extension.

The definition of the extension shall contain a (possibly empty) list of attribute declarations for every component. An attribute declaration shall contain the *key* of the attribute, the *datatype* of the attribute, and a (possible empty) list of *aliases*. An alias shall contain the *descriptive text* for the attribute (that is, the commonly understood semantics for the attribute) and the *language code* of the language of this descriptive text.

5. XES XML serialization

Figure 1 presents an overview of the XES metadata structure. An explanation of the elements follows in this clause. Annex D shows the (normative) schema definition for XES. Figure 2 is a flow diagram for the XES XML serialization state machine.

5.1 Log element

Captures the log component from the XES metadata structure.

— XML name: *log*.

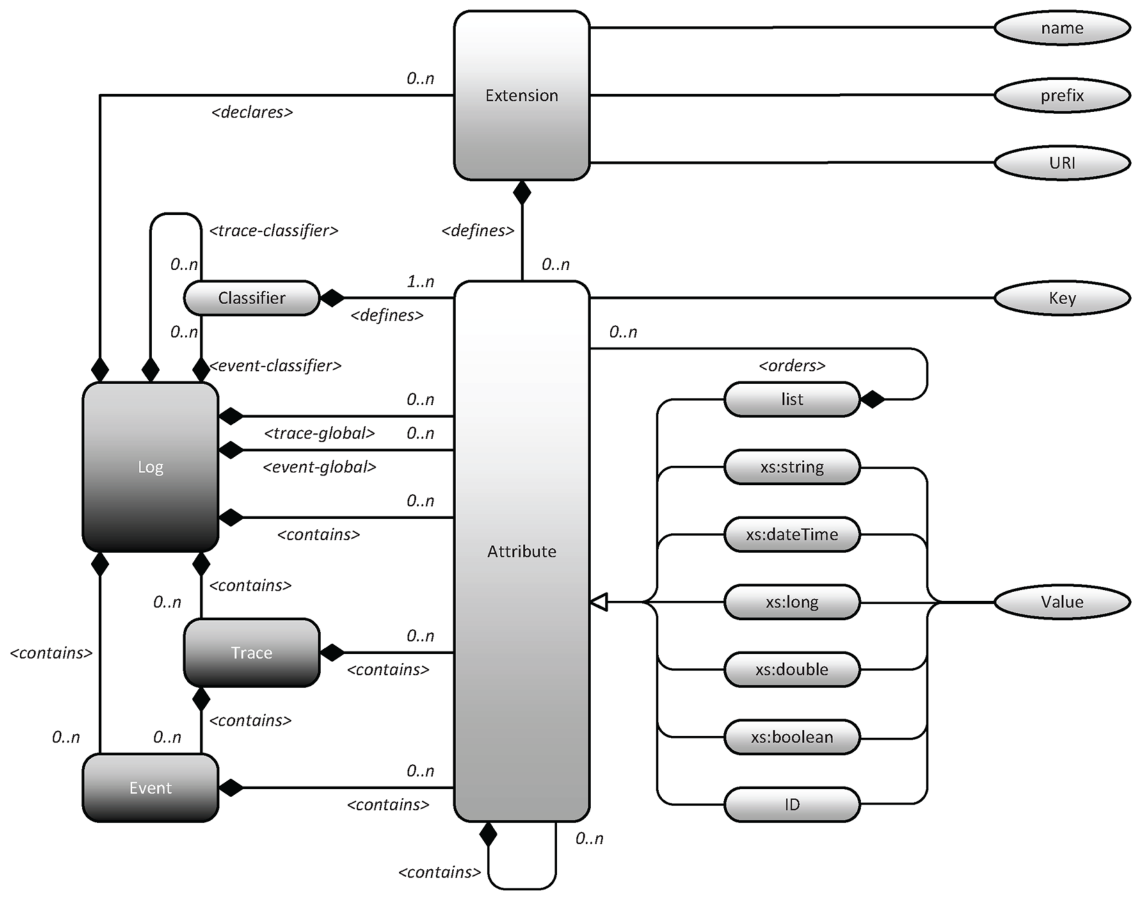

Figure 1—Overview of the XES metadata structure

5.1.1 Elements

The following (sub) elements shall appear in the specified order.

XML name	*XES element*	*Min*	*Max*	*Description*
extension	Extension, see 5.5.	0	∞	An extension declaration for the log.
global	Global, see 5.6.	0	∞	A list of global (event or trace) attributes for the log.
classifier	Classifier, see 5.7.	0	∞	A classifier (event or trace) definition for the log.
attribute	Attribute, see 5.4.	0	∞	An attribute for the log. Shall be elementary or composite.
trace	Trace, see 5.2.	0	∞	A trace for the log.
event	Event, see 5.3.	0	∞	An event for the log.

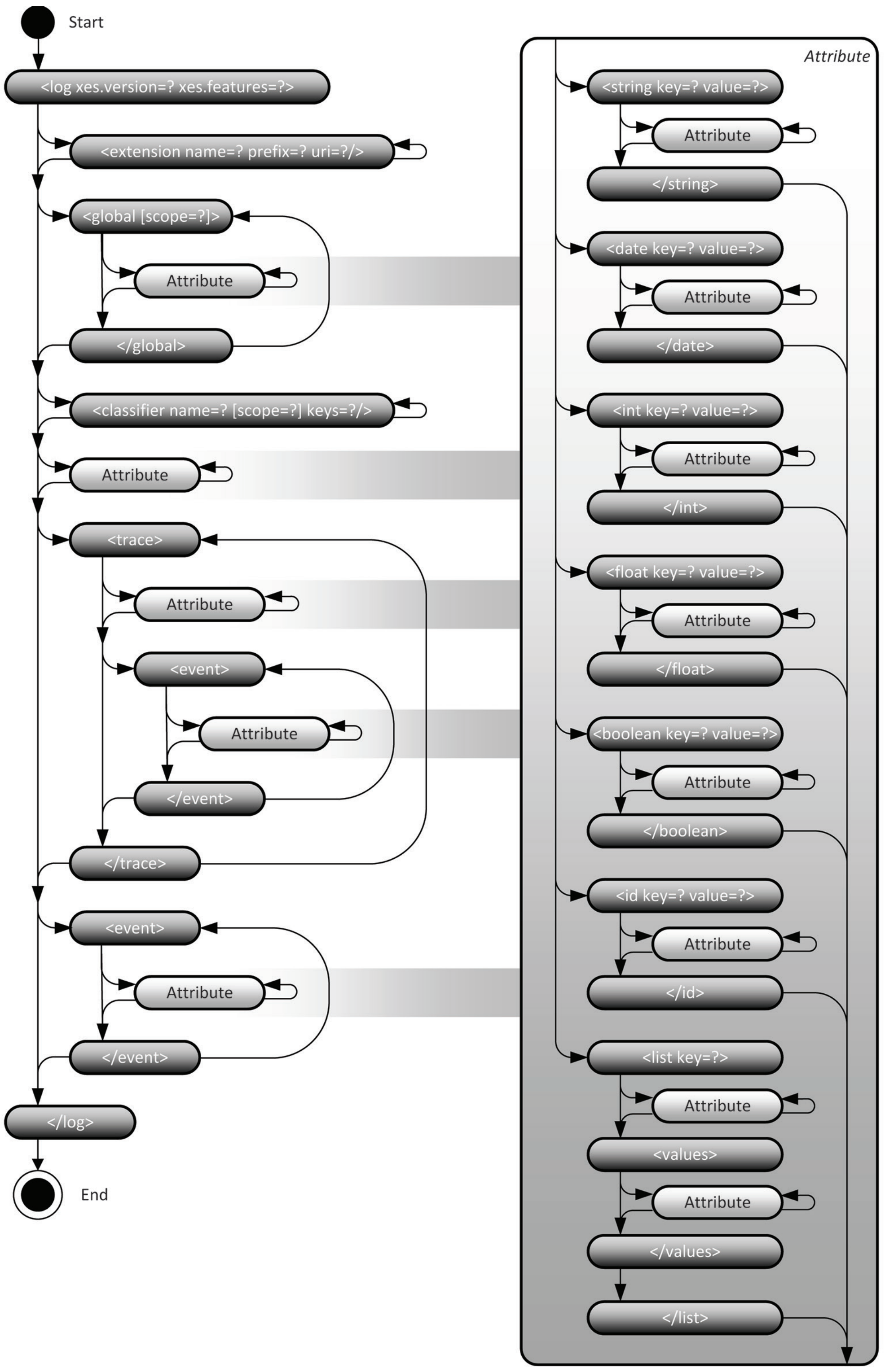

Figure 2—State machine flow diagram for XES XML serialization

5.1.2 Attributes

Attribute key	***Attribute type***	***Status***	***Description***
xes.version	*xs:decimal*	Required	The version of the XES standard the document conforms to (e.g., 2.0).
xes.features	*xs:token*	Required	A whitespace-separated list of optional XES features this document makes use of (e.g., *nested-attributes*). If no optional features are used, this attribute shall have an empty value.

5.2 Trace element

Captures the trace component from the XES metadata structure.

— XML name: *trace*.

5.2.1 Elements

The following (sub) elements shall appear in the specified order.

XML name	***XES element***	***Min***	***Max***	***Description***
attribute	Attribute, see 5.4.	0	∞	An attribute for the trace. Shall be elementary or composite.
event	Event, see 5.3.	0	∞	An event for the trace.

5.2.2 Attributes

N/A.

5.3 Event element

Captures the event component from the XES metadata structure.

— XML name: *event*.

5.3.1 Elements

The following (sub) elements shall appear in the specified order.

XML name	***XES element***	***Min***	***Max***	***Description***
attribute	Attribute, see 5.4.	0	∞	An attribute for the event. Shall be elementary or composite.

5.3.2 Attributes

N/A.

5.4 Attribute element

Captures the attribute component from the XES metadata structure. Shall be any of the following:

XML name	***Datatype***	***Value***	***Description***
string	*xs:string*	Elementary	A string value.
date	*xs:dateTime*	Elementary	A UTC time value.
int	*xs:long*	Elementary	An integer number value.
float	*xs:double*	Elementary	A real number value.
boolean	*xs:boolean*	Elementary	A Boolean value.
id	ID	Elementary	A UUID value.
list	Attribute list	Composite	A sorted list of attributes.

5.4.1 Elements for elementary attributes

The following (sub) elements shall appear in the specified order.

XML name	***XES element***	***Min***	***Max***	***Description***
attribute	Attribute, see 5.4.	0	∞	An attribute for the attribute, that is, a meta-attribute. Shall be elementary or composite.

5.4.2 Elements for composite attributes

The following (sub) elements shall appear in the specified order.

XML name	***XES element***	***Min***	***Max***	***Description***
attribute	Attribute, see 5.4.	0	∞	An attribute for the attribute, that is, a meta-attribute. Shall be elementary or composite.
values	Attribute list	1	1	The ordered list of attributes that constitute the composite value of the attribute. Attributes in this list shall be elementary or composite, and attributes in this list may share the same key.

5.4.3 Attributes for elementary attributes

Attribute key	***Attribute type***	***Status***	***Description***
key	*xs:string*	Required	The key of the attribute.
value	*xs:string*	Required	The value (as a string) of the elementary attribute.

5.4.4 Attributes for composite attributes

Attribute key	***Attribute type***	***Status***	***Description***
key	*xs:string*	Required	The key of the attribute.

5.5 Extension element

Captures the extension declaration from the XES metadata structure.

— XML name: *extension*.

5.5.1 Elements

N/A.

5.5.2 Attributes

Attribute key	***Attribute type***	***Status***	***Description***
name	*xs:NCName*	Required	The name of the extension.
prefix	*xs:NCName*	Required	The prefix of the extension.
uri	*xs:anyURI*	Required	The URI from where the definition of this extension (shall be a file that conforms to the XESEXT format) can be retrieved.

5.6 Global element

Captures the global attribute declaration from the XES metadata structure.

— XML name: *global*.

5.6.1 Elements

XML name	***XES element***	***Min***	***Max***	***Description***
attribute	Attribute, see 5.4.	0	∞	A global attribute declaration for the log. Depending on whether the attribute is declared global for events or traces, every event or trace in the log shall have this attribute as sub element.

The attribute value provided with this global declaration shall only be used if a new event or trace needs to be inserted in the log for which no other valid value for this attribute can be obtained. In that case, and only in that case, the value of this attribute may be used.

5.6.2 Attributes

Attribute key	***Attribute type***	***Status***	***Description***
scope	*xs:NCName*	Optional	Shall be either *event* or *trace*, to denote whether this attribute is declared global for events or traces. The default is *event*.

5.7 Classifier element

Captures the classifier definition from the XES metadata structure.

— XML name: *classifier*.

5.7.1 Elements

N/A.

5.7.2 Attributes

Attribute key	*Attribute type*	*Status*	*Description*
name	*xs:NCName*	Required	The name of the classifier.
scope	*xs:NCName*	Optional	Shall be either *event* or *trace*, to denote whether this classifier shall be used to classify events or traces. The default is *event*.
keys	*xs:token*	Required	The white-space-separated list of attribute keys that constitute this classifier. These attributes shall be declared global at the proper (either *event* or *trace*) level.

6. XESEXT XML Serialization

6.1 XES Extension element

The XES Extention element (XESEXT) captures the extension definition from the XES metadata structure.

— XML name: *xesextension*.

Figure 3 shows the XESEXT XML serialization state machine diagram. Annex E shows the (normative) schema definition for XESEXT. Annex F shows an informative example of the XESEXT XML serialization.

6.1.1 Elements

The following (sub) elements shall appear in the specified order.

XML name	*XESEXT element*	*Min*	*Max*	*Description*
log	Log, see 6.2.	0	1	Attribute definitions for logs.
trace	Trace, see 6.3.	0	1	Attribute definitions for traces.
event	Event, see 6.4	0	1	Attribute definitions for events.
meta	Meta, see 6.5	0	1	Attribute definitions for attributes.

6.1.2 Attributes

Attribute key	*Attribute type*	*Status*	*Description*
name	*xs:NCName*	Required	The name of the extension.
prefix	*xs:NCName*	Required	The prefix to be used for this extension.
uri	*xs:anyURI*	Required	The URI where this extension can be retrieved from.

6.2 Log element

Captures the log extension definition from the XES metadata structure.

— XML name: *log*.

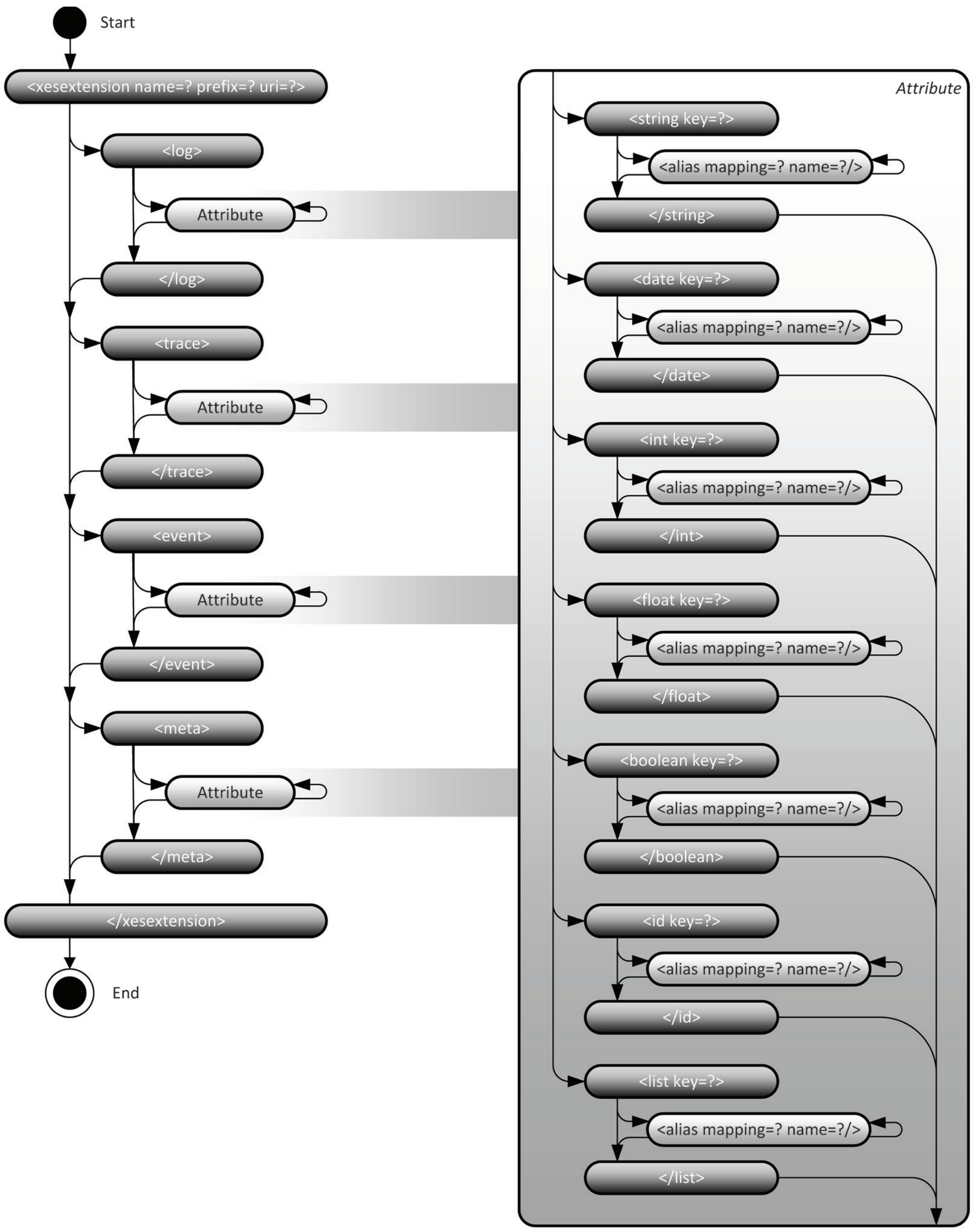

Figure 3—State machine diagram for XESEXT XML serialization

6.2.1 Elements

The following (sub) elements shall appear in the specified order.

XML name	*XESEXT element*	*Min*	*Max*	*Description*
attribute	Attribute, see 6.6.	0	∞	Attribute definition for logs.

6.2.2 Attributes

N/A.

6.3 Trace element

Captures the trace extension definition from the XES metadata structure.

— XML name: *trace*.

6.3.1 Elements

The following (sub) elements shall appear in the specified order.

XML name	***XESEXT element***	***Min***	***Max***	***Description***
attribute	Attribute, see 6.6.	0	∞	Attribute definition for traces.

6.3.2 Attributes

N/A.

6.4 Event element

Captures the event definition from the XES metadata structure.

— XML name: *event*.

6.4.1 Elements

The following (sub) elements shall appear in the specified order.

XML name	***XESEXT element***	***Min***	***Max***	***Description***
attribute	Attribute, see 6.6.	0	∞	Attribute definition for events.

6.4.2 Attributes

N/A.

6.5 Meta element

Captures the meta (attribute) extension definition from the XES metadata structure.

— XML name: *meta*.

6.5.1 Elements

The following (sub) elements shall appear in the specified order.

XML name	***XESEXT element***	***Min***	***Max***	***Description***
attribute	Attribute, see 6.6.	0	∞	Attribute definition for attributes.

6.5.2 Attributes

N/A.

6.6 Attribute element

Captures the attribute extension element from the XES metadata structure. Shall be any of the following:

XML name	*Description*
string	A string valued attribute.
date	A UTC time valued attribute.
int	An integer number valued attribute.
float	A real number valued attribute.
boolean	A Boolean valued attribute.
id	A UUID valued attribute.
list	A list valued attribute.

6.6.1 Elements

The following (sub) elements shall appear in the specified order.

XML name	*XESEXT element*	*Min*	*Max*	*Description*
alias	Attribute, see 6.7.	0	∞	Aliases for this attribute.

6.6.2 Attributes

Attribute key	*Attribute type*	*Status*	*Description*
key	*xs:string*	Required	The key of the attribute.

6.7 Alias

Captures the alias extension element from the XES metadata structure.

6.7.1 Elements

N/A.

6.7.2 Attributes

Attribute key	*Attribute type*	*Status*	*Description*
mapping	*xs:NCName*	Required	The language code (using the ISO 639-1 and ISO 639-2 standards) for this alias.
name	*xs:string*	Required	The semantics of this attribute described using the language with the given code.

7. XES standard extensions

XES shall recognize and treat all extensions as equal, independent from their source. This allows users of the format to extend it, in order to fit any purpose or domain setting. However, there are recurring requirements for information stored in event logs, which demand a fixed and universally understood semantics. For this purpose, a number of extensions have been standardized. When creating logs for a specific domain, or also when designing log-analyzing techniques, one should consider using these standardized extensions, since they allow for a wider level of understanding of the contents of event logs.

In the following subclauses, the currently standardized extensions to the XES formats are introduced.

7.1 Concept extension

The Concept extension defines, for all levels of the XES type hierarchy, an attribute which stores the generally understood name of type hierarchy elements.

Name	***Prefix***	***URI***
Concept	*concept*	http://www.xes-standard.org/concept.xesext

7.1.1 Attributes

Name	***Key***	***Components***	***Datatype***	***Description***
Name	*name*	Log, Trace, Event	*xs:string*	Stores a generally understood name for any component type. For streams and logs, the name attribute may store the name of the process having been executed. For traces, the name attribute usually stores the case ID. For events, the name attribute represents the name of the event, e.g., the name of the executed activity represented by the event.
Instance	*instance*	Event	*xs:string*	This represents an identifier of the activity instance whose execution has generated the event. This way, multiple instances (occurrences) of the same activity can be told apart.

7.2 Lifecycle extension

The Lifecycle extension specifies for events the lifecycle transition they represent in a transactional model of their generating activity. This transactional model can be arbitrary. However, the Lifecycle extension also specifies two standard transactional models for activities, see the figures that follow. The use of this extension is appropriate in any setting where events denote lifecycle transitions of higher-level activities.

Name	***Prefix***	***URI***
Lifecycle	*lifecycle*	http://www.xes-standard.org/lifecycle.xesext

7.2.1 Attributes

Name	*Key*	*Components*	*Datatype*	*Description*
Model	*model*	Log	*xs:string*	This attribute refers to the lifecycle transactional model used for all events in the log. If this attribute has a value of *standard*, the standard lifecycle transactional model of this extension is assumed. If it is has a value of *bpaf*, the Business Process Analytics Format (BPAF) lifecycle transactional model is assumed.
Transition	*transition*	Event	*xs:string*	The transition attribute is defined for events, and specifies the lifecycle transition of each event.
State	*state*	Event	*xs:string*	The state attribute is defined for events, and specifies the lifecycle state of each event.

7.2.2 BPAF lifecycle transactional model

Note that the transitions shown in Figure 4 are the most typical transitions, but manual interventions and different system implementations may lead to additional transitions not depicted in the model.

State	*Description*
Closed	The activity is closed for execution.
Closed.Cancelled	The execution of the activity is cancelled.
Closed.Cancelled.Aborted	The execution of the activity is aborted.
Closed.Cancelled.Error	The execution of the activity is in error.
Closed.Cancelled.Exited	The execution of the activity is exited manually.
Closed.Cancelled.Obsolete	The execution of the activity is obsolete (e.g., in case of a timeout).
Closed.Cancelled.Terminated	The execution of the activity is forcibly terminated.
Completed	The execution of the activity is completed, i.e., it has ended naturally.
Completed.Failed	The execution of the activity is completed, but the result is unsuccessful (from a business perspective).
Completed.Success	The execution of the activity is completed, and the result is successful (from a business perspective).
Open	The activity is open for execution.
Open.NotRunning	The execution of the activity is not on-going.
Open.NotRunning.Assigned	The activity is on the worklists of resources.
Open.NotRunning.Reserved	The activity has been selected by a resource to work on.
Open.NotRunning.Suspended.Assigned	The activity is on the worklists of resources, but is barred from execution.
Open.NotRunning.Suspended.Reserved	The activity has been selected by a resource, but is barred from execution.
Open.Running	The execution of the activity is on-going.
Open.Running.InProgress	The execution of the activity is in progress.
Open.Running.Suspended	The execution of the activity is on-going, but not in progress.

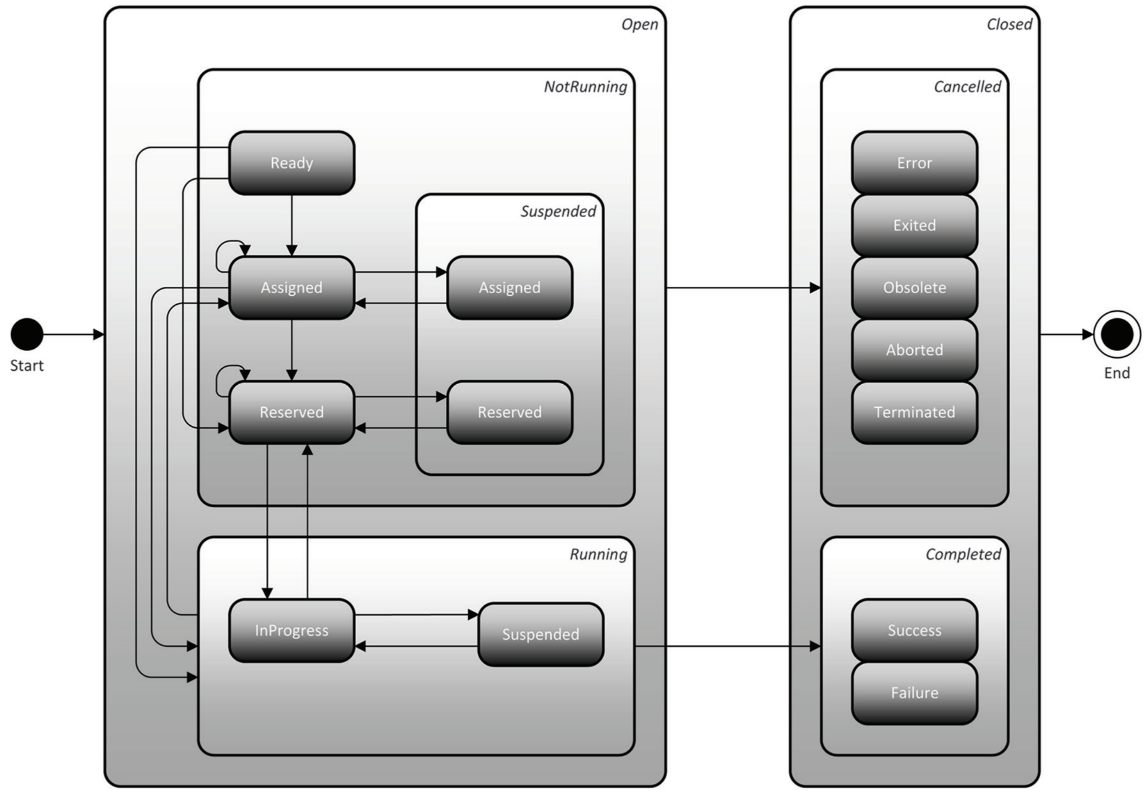

Figure 4—State machine for the BPAF transactional model

7.2.3 Standard lifecycle transition model

In contrast with the BPAF transactional model, the standard lifecycle model (Figure 5) uses the transitions instead of the states to denote the new state of an activity.

Transition	***From State***	***To State***	***Description***
assign	Open.NotRunning.Ready	Open.NotRunning.Assigned	The activity is assigned for execution to a resource.
ate_abort	Open.Running	Closed.Cancelled.Aborted	The execution of the activity is aborted, the execution of the corresponding case is not aborted. As a result, execution of the activity has failed.
autoskip	Start	Closed. Cancelled. Obsolete	The execution of the activity is skipped by the system. As a result, execution of the activity has succeeded.
complete	Open.Running. InProgress	Closed. Completed	The execution of the activity is completed. As a result, execution of the activity has succeeded.
manualskip	Open.NotRunning	Closed. Cancelled. Obsolete	The execution of the activity is skipped by the user. As a result, execution of the activity has succeeded.
pi_abort	Open	Closed.Cancelled.Aborted	The execution of the activity is aborted, the execution of the corresponding case is also aborted. As a result, execution of the activity has failed.
reassign	Open.NotRunning.Assigned	Open.NotRunning.Assigned	The activity has been reassigned for execution to another resource

Transition	***From State***	***To State***	***Description***
resume	Open.Running.Suspended	Open.Running InProgress	The execution of the activity is resumed.
schedule	Start	Open.NotRunning.Ready	The activity is scheduled for execution.
start	Open.NotRunning.Assigned	Open.Running.InProgress	The execution of the activity is started.
suspend	Open.Running.InProgress	Open.Running.Suspended	The execution of the activity is suspended.
unknown	Any	Any	Any lifecycle transition not captured by any of the other transitions.
withdraw	Open.NotRunning	Closed.Cancelled.Exited	The assignment of the activity is revoked. As a result, execution of the activity has failed.

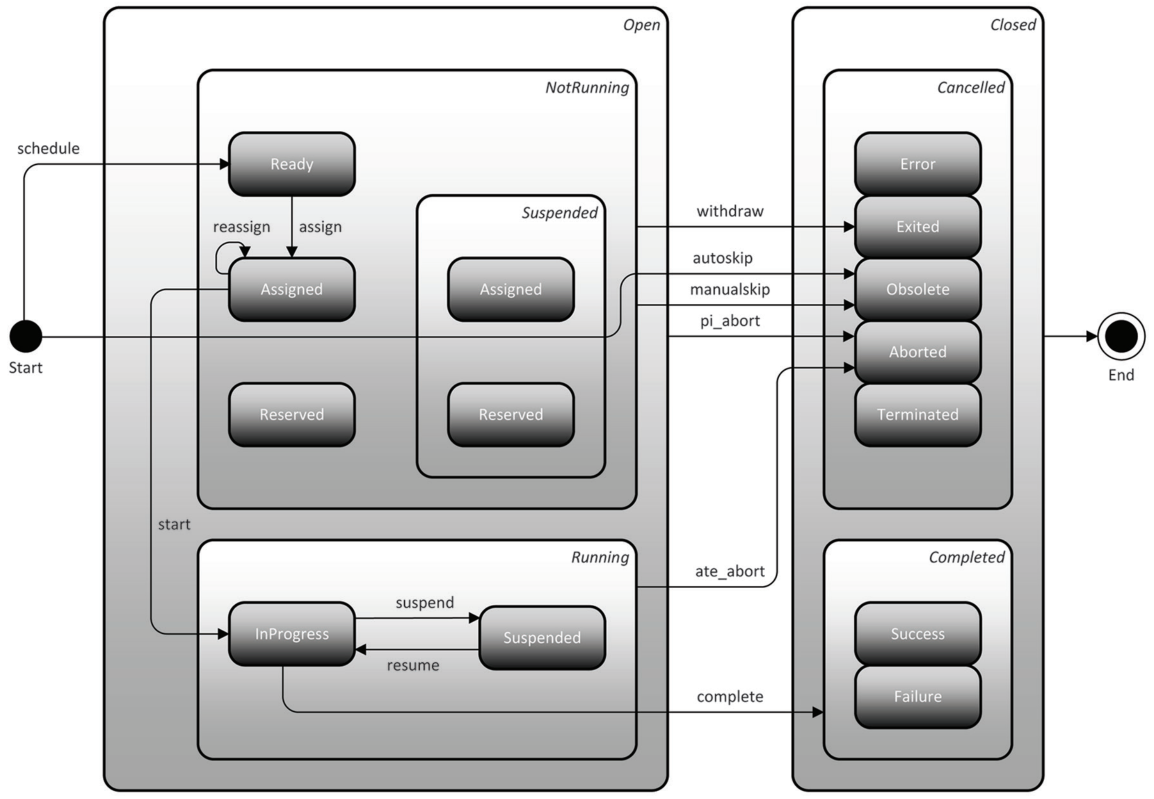

Figure 5—State machine for the standard transactional model

7.3 Organizational extension

The organizational extension is useful for domains, where events can be caused by human actors, who are somewhat part of an organizational structure. This extension specifies three attributes for events, which identify the actor having caused the event, and his position in the organizational structure.

Name	***Prefix***	***URI***
Organizational	*org*	http://www.xes-standard.org/org.xesext

7.3.1 Attributes

Name	*Key*	*Components*	*Datatype*	*Description*
Resource	*resource*	Event	*xs:string*	The name, or identifier, of the resource that triggered the event.
Role	*role*	Event	*xs:string*	The role of the resource that triggered the event, within the organizational structure.
Group	*group*	Event	*xs:string*	The group within the organizational structure, of which the resource that triggered the event is a member.

7.4 Time extension

In almost all applications, the exact date and time at which events occur can be precisely recorded. Storing this information is the purpose of the time extension. Recording a UTC time for events is important, since this constitutes crucial information for many event log analysis techniques.

Name	*Prefix*	*URI*
Time	*time*	http://www.xes-standard.org/time.xesext

7.4.1 Attributes

Name	*Key*	*Components*	*Datatype*	*Description*
Time	*timestamp*	Event	*xs:dateTime*	The UTC time at which the event occurred.

7.5 Semantic extension

Depending on the view on a process, type hierarchy artifacts may correspond to different concepts. For example, the name of an event (as specified by the Concept extension) may refer to the activity whose execution has triggered this event. However, this activity may be situated on a low level in the process meta-model, and be a part of higher-level, aggregate activities itself.

Besides events, also other elements of the XES type hierarchy may refer to a number of concepts at the same time (e.g., a log may refer to different process definitions, on different levels of abstractions). To express the fact, that one type artifact may represent a number of concepts in a process meta-model, the semantic extension has been defined.

It is assumed that there exists an ontology for the process meta-model, where every concept can be identified by a unique URI. The semantic extension defines an attribute, which allows to store a number of model references, as URIs, in any element of the XES type hierarchy.

Name	*Prefix*	*URI*
Semantic	*semantic*	http://www.xes-standard.org/semantic.xesext

7.5.1 Attributes

Name	*Key*	*Components*	*Datatype*	*Description*
Model reference	*modelReference*	Log, Trace, Event, Meta	*xs:string*	References to model concepts in an ontology. Model References are stored in a literal string, as comma-separated URIs identifying the ontology concepts.

7.6 ID extension

The ID extension provides unique identifiers (UUIDs) for elements.

Name	*Prefix*	*URI*
Identity	*identity*	http://www.xes-standard.org/identity.xesext

7.6.1 Attributes

Name	*Key*	*Components*	*Datatype*	*Description*
Id	*id*	Log, Trace, Event, Meta	ID	Unique identifier (UUID) for an element.

7.7 Cost extension

The cost extension defines a nested element to store information about the cost associated with activities within a log. The objective of this extension is to provide semantics to cost aspects that can be associated with events in a log. The definition associates three data elements with a particular cost element: the amount associated with the cost element as well as the cost driver that is responsible for incurring that cost and the cost type. As it is possible for more than one cost element to be associated with an event, the cost incurred per event is summarized using the total attribute. The currency element is also recorded once per event. Cost information can be recorded at the trace level (for instance, to be able to say that it costs $20 when a case is started). Cost information can also be recorded at the event level (for instance, for certain event types such as complete or canceled events) to capture the cost incurred in undertaking the activity by a resource.

Name	*Prefix*	*URI*
Cost	*cost*	http://www.xes-standard.org/cost.xesext

7.7.1 Attributes

Name	*Key*	*Components*	*Datatype*	*Description*
Total	*total*	Trace, Event	*xs:double*	Total cost incurred for a trace or an event. The value represents the sum of all the cost amounts within the element.
Currency	*currency*	Trace, Event	*xs:string*	The currency (using the ISO 4217:2008 standard) of all costs of this element.
Drivers	*drivers*	Trace, Event	List	A detailed list containing cost driver details.

Name	***Key***	***Components***	***Datatype***	***Description***
Amount	*amount*	Meta	*xs:double*	The value contains the cost amount for a cost driver.
Driver	*driver*	Meta	*xs:string*	The value contains the id for the cost driver.
Type	*type*	Meta	*xs:string*	The value contains the cost type (e.g., *Fixed, Overhead, Materials*).

The drivers attribute shall contain any number of driver attributes, and every driver attribute shall contain the amount and type attribute, like follows:

```
<event>
  <string key="cost:currency" value="AUD" />
  <string key="cost:total" value="123.50" />
  <list key="cost:drivers">
    <values>
      <string key="driver" value="d2f4ee27">
        <float key="amount" value="21.40" />
        <string key="type" value="Labour" />
      </string>
      <string key="driver" value="abc124">
        <float key="amount" value="102.10" />
        <string key="type" value="Variable Overhead" />
      </string>
    </values>
  </list>
</event>
```

8. XES Conformance

Conformance to the XES part of this standard is discussed in 8.1 and 8.2. In 8.1 and 8.2, *strictly conforming XES XML instance* and *conforming XES XML instance* refer to the metadata represented in the XES XML instance before any processing of the XES XML instance.

8.1 Strictly conforming XES instances

A strictly conforming XES instance shall consist solely of XES data elements as defined in Clause 4, and shall conform to the requirements of Clause 4. A strictly conforming XES XML instance

— Shall be a strictly conforming XES instance as defined before

— Shall conform to the requirements of Clause 5

— May not include XML elements or attributes that are not defined in Clause 5

— May not include mixed content

8.2 Conforming XES instances

A conforming XES instance may contain XES data elements as defined in Clause 4, and shall conform to the requirements of Clause 4. A conforming XES XML instance

— Shall be a conforming XES instance as defined before

— Shall conform to the requirements of Clause 5

— May include XML elements or attributes that are not defined in Clause 5

— May include mixed content

9. XESEXT Conformance

Conformance to the XESEXT part of this standard is discussed in 9.1 and 9.2. In 9.1 and 9.2, *strictly conforming XESEXT XML instance* and *conforming XESEXT XML instance* refer to the metadata represented in the XESEXT XML instance before any processing of the XESEXT XML instance.

9.1 Strictly conforming XESEXT instances

A strictly conforming XESEXT instance shall consist solely of XESEXT data elements as defined in 4.5, and shall conform to the requirements of 4.5. A strictly conforming XESEXT XML instance

— Shall be a strictly conforming XES instance as defined before

— Shall conform to the requirements of Clause 6

— May not include XML elements or attributes that are not defined in Clause 6

— May not include mixed content

9.2 Conforming XESEXT instances

A conforming XESEXT instance may contain XESEXT data elements as defined in 4.5, and shall conform to the requirements of 4.5. A conforming XES XML instance

— Shall be a conforming XES instance as defined before

— Shall conform to the requirements of Clause 6

— May include XML elements or attributes that are not defined in Clause 6

— May include mixed content

Annex A

(informative)

History of XES

Unlike classical process analysis tools, which are purely model-based (like simulation models), process mining requires event logs. Fortunately, today's systems provide detailed event logs. Process mining has emerged as a way to analyze systems (and their actual use) based on the event logs they produce [B1], [B11], [B23], [B24], [B25], [B32].[7] Note that, unlike classical data mining, the focus of process mining is on concurrent processes and not on static or mainly sequential structures. Also note that commercial business intelligence (BI) tools are not doing any process mining. They typically look at aggregate data seen from an external perspective (including frequencies, averages, utilization levels, and service levels). Unlike BI tools, process mining looks "inside the process" and allows for insights at a much more refined level.

The omnipresence of event logs is an important enabler of process mining, as analysis of run-time behavior is only possible if events are recorded. Fortunately, all kinds of information systems provide such logs, which include classical workflow management systems like FileNet and Staffware, ERP systems like SAP, case handling systems like BPM|one, PDM systems like Windchill, CRM systems like Microsoft Dynamics CRM, and hospital information systems like Chipsoft.[8] These systems provide very detailed information about the activities that have been executed.

However, also all kinds of embedded systems increasingly log events. An embedded system is a special-purpose system in which the computer is completely encapsulated by or dedicated to the device or system it controls. Examples include medical systems like X-ray machines, mobile phones, automobile entertainment systems, production systems like wafer steppers, copiers, and sensor networks. Software plays an increasingly important role in such systems and, already today, many of these systems log events. An example is the "CUSTOMerCARE Remote Services Network" of Philips Medical Systems (PMS), which is a worldwide internet-based private network that links PMS equipment to remote service centers.[9] Any event that occurs within an X-ray machine (like moving the table or setting the deflector) is recorded and can be analyzed remotely by PMS. The logging capabilities of the machines of PMS illustrate the way in which embedded systems produce event logs.

The MXML format [B29] has proven its use as a standard event log format in process mining. However, based on practical experiences with applying MXML in about one hundred organizations, several problems and limitations related to the MXML format have been discovered. One of the main problems is the semantics of additional attributes stored in the event log. In MXML, these are all treated as string values with a key and have no generally understood meaning. Another problem is the nomenclature used for different concepts. This is caused by MXML's assumption that strictly structured processes would be stored in this format [B8].

To solve the problems encountered with MXML and to create a standard that could also be used to store event logs from many different information systems directly, a new event log format was developed. This new event log format is named XES, which stands for eXtensible Event Stream [B30]. This XES standard has been adopted for standardization by the IEEE Task Force Process Mining [B8].

[7] The numbers in brackets correspond to those of the bibliography in Annex G.
[8] This information is given for the convenience of users of this standard and does not constitute an endorsement by the IEEE of these products. Equivalent products may be used if they can be shown to lead to the same results.
[9] See Footnote 8.

Annex B

(informative)

Current status of XES

At the moment, there already exists a XES standard [B32] (which has not been endorsed by the IEEE), that comes with a reference implementation called OpenXES [B14]. The IEEE XES Standard differs from the XES 2.0 standard in the following respects (see also [B2]):

— *Events in logs:* In the XES 2.0 Standard, every event has to be contained in some trace. In the IEEE XES Standard, events may also be contained by the log itself.

— *Classifiers use ordered attribute keys:* In the XES 2.0 standard, a classifier corresponds to a set of attribute keys. In the IEEE XES Standard, a classifier corresponds to a list of attribute keys.

— *Trace classifier:* In the XES 2.0 Standard, only event classifiers are defined. In the IEEE XES Standard, a trace classifier that allows classification of an entire trace is also defined. This can be useful in case the events in that trace lack the attributes for a corresponding event classifier.

— *List attribute values:* In the XES 2.0 Standard, a list cannot have any metadata as all attributes of the list are considered to be values of this list. In the IEEE XES Standard, there is a new element called values to hold the list values, which allows for the list metadata.

— *Container attributes:* In the XES 2.0 Standard, a container attribute is defined. In the IEEE XES Standard, this attribute has been dropped. The list attribute suffices.

— *Lifecycle extension:* In the XES 2.0 Standard, only the transition labels are defined. In the IEEE XES Standard, the state labels (as introduced by [B9]) are also used.

— *Timestamps:* In the XES 2.0 Standard, timestamps can be local timestamps; that is, they are allowed to lack relevant time zone information. In the IEEE XES Standard, timestamps are required to in UTC time, which enforces either using the UTC time zone or using a time offset to the UTC time zone.

Annex C

(informative)

XES support

C.1 Tool support

The latest version of the XES Standard, 2.0, is supported by many tools including:[10]

— AProMore [B2]: an advanced process model repository.

— Celonis Process Mining [B3]: a process mining tool that retrieves and visualizes all of the process data saved in your IT systems.

— CoBeFra [B22]: a comprehensive benchmarking framework for conformance checking.

— CoPrA Tool [B4]: a tool for communication analysis and process mining of team processes.

— Disco [B5]: a process mining tool that helps you to discover your processes.

— Flipflop [B6]: a test and Flip net synthesis tool for the automated synthesis of surgical procedure models.

— JIRAVIEW [B10]: a tool to extract data from JIRA through REST and create charts.

— Lean Document Production toolkit [B15]: a toolkit to model and analyze complex print operations.

— minit [B12]: a modern process mining tool for complex process discovery, visualization and analysis.

— OpenXES [B13]: the XES reference implementation, an Open Source Java library for reading, storing, and writing XES logs.

— PMLAB [B21]: a scripting environment for Process Mining in Python, which allows to perform exploratory process-oriented computing and/or research in a process-oriented language.

— ProM 6 [B14]: an extensible Process Mining framework, which is basically the breeding ground for many new process mining related techniques.

— RapidProM [B16]: a ProM 6 Framework extension to RapidMiner 5.3.

— Rialto PI (Process Intelligence): The Rialto suite [B17] from Exeura covers Data Mining, Text Mining, Reasoning and Process Mining in one single environment.

— ruby-xes [B19]: a Ruby library for generating XES event log.

— XES Python Tool [B31]: a simple Python tool for generating XES files for Process Mining.

— XESame: a tool for extracting XES logs from databased, distributed with ProM 6.

— YAWL [B33]: a BPM/Workflow system, based on a concise and powerful modeling language, that handles complex data transformations, and full integration with organizational resources and external Web Services.

C.2 Data support

A number of data sets have been published that use the XES standard:

[10]This information is given for the convenience of users of this standard and does not constitute an endorsement by the IEEE of these products. Equivalent products may be used if they can be shown to lead to the same results.

— A real-life log from five Dutch municipalities containing all building permit applications over a period of approximately five years [B27].

— A real-life log from a Dutch financial institute that contains some 262.200 events in 13.087 cases [B26].

— A real-life log from a Dutch academic hospital that contains some 150.000 events in over 1100 cases [B28].

— A real-life log from Volvo IT Belgium [B20]. The log contains events from an incident and problem management system called VINST.

These data sets are publicly available and can be used as benchmark data sets for many computational intelligence techniques. The use of the DOIs makes it easy to refer to the data sets.

C.3 Publication support

The XES standard has appeared in a number of publications:

— Accorsi, R., C. Wonnemann, and S. Dochow, "SWAT: A Security Analysis Toolkit for Reliably Process-aware Information Systems," *Sixth International Conference on Availability, Reliability, and Security (ARES)*, pp. 692–697, 2011.

— Accorsi, R., T. Stocker, and G. Müller, "On the exploitation of process mining for security audits: the process discovery case," *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, pp. 1462–1468, 2013.

— Aruna Devi, C and Sudhamani, "Application of business process mining using control flow perspective in manufacturing unit," *International Journal of Emerging Trends and Technology in Computer Science*, vol. 2, no. 5, pp. 130–133, 2013.

— Baumgrass, A., "Deriving Current-State RBAC Models from Event Logs." *Sixth International Conference on Availability, Reliability, and Security (ARES)*, pp. 667–672, 2011.

— Becker J., M. Matzner, O. Müller, and M. Walter, "A Review of Event Formats as Enablers of Event-Driven BPM," *Lecture Notes in Business Information Processing*, vol. 99, pp. 433–445, 2012.

— Bernardi, M.L., M. Cimitile, C. Di Francescomarino, and F.M.Maggi, "Using Discriminative Rule Mining to Discover Declarative Process Models with Non-atomic Activities," *Lecture Notes in Computer Science*, vol. 8620, pp. 281–295, 2014.

— Bose, R.P.J.C., F.M. Maggi, and W.M.P. van der Aalst, "Enhancing Declare Maps Based on Event Correlations." *Lecture Notes in Computer Science*, vol. 8094, pp. 97–112, 2013.

— Bui, D.B., Dazic, F., and Hecker, M., "Application of Tree-structured Data Mining for Analysis of Process Logs in XML format." In *AusDM 2012*, CR-PIT 134, edited by Y. Zhao et al., 109–118. Australian Computer Society, 2012.

— Burattin, A., *Process Mining Techniques in Business Environments: Theoretical Aspects, Algorithms, Techniques and Open Challenges in Process Mining*. Heidelberg: Springer-Verlag, 2015.

— Burattin, A., F.M. Maggi, and A. Sperduti, "Conformance checking based on multi-perspective declarative process models," *Expert Systems with Applications*, vol. 65, pp. 194–211, December 2015.

— Caron, F. et al., "A Process Mining-based Investigation of Adverse Events in Care Processes." *Health Information Management Journal*, vol. 43, no. 1, pp. 16–25, 2014.

— Dolean, C.-C., "Mining Product Data Models: A Case Study." *Informatica Economică*, vol. 18, no. 1, pp. 69–82, 2014.

— Engel, R., J.R. Prabhakara, C. Pichler, M. Zapletal, and H. Werthner, "EDIminer: A Toolset for Process Mining from EDI Messages." *25th International Conference on Information Systems Engineering*, pp. 146–153, 2013.

— Engel, R., W. Krathu, M. Zapletal, C. Pichler, W.M.P. van der Aalst, and H. Werthner, "Process Mining for Electronic Data Interchange." *Lecture Notes in Business Information Processing*, vol. 85, pp. 77–88, 2011.

— Gontar, B. and Z. Gontar, "Event Log Standards in BPM Domain." *Information Systems in Management*, vol. 1, no. 4, pp. 293–304, 2012.

— Günther, C.W., "Process Mining in Flexible Environments." PhD thesis, Eindhoven University of Technology, 2009 [B7].

— Jones, W., "The Future of Personal Information Management, Part 1: Our Information, Always and Forever." *Synthesis Lectures on Information Concepts, Retrieval, and Services*, vol. 4, no. 1, 1–125, 2012.

— Kaes, G., S. Rinderle-Ma, R. Vigne, and J. Mangler, "Flexibility Requirements in Real-World Process Scenarios and Prototypical Realization in the Care Domain." *Lecture Notes in Computer Science*, vol. 8842, pp. 55–64, 2014.

— Khoadandelou, G., C. Hug, R. Deneckère, and C. Salinesi, "Process Mining *Versus* Intention Mining." *ALecturer Notes in Business Information Processing,* vol. 147, pp. 466–480, 2013.

— Krinkin, K., E. Kalishenko, and S.P.S. Prakash, "Process Mining Approach for Traffic Analysis in Wireless Mesh Networks." *Lecture Notes in Computer Science*, vol. 7469, pp. 260–269, 2012.

— Le, N.T.T., H. Chihab, S. Stinckwich, and T.V. Ho, "Representing, Simulating and Analysing Ho Chi Minh City Tsunami Plan by Means of Process Models," Information Systems for Crisis Response and Management Conference, 2013. Available at https://arxiv.org/abs/1312.4851v1.

— Ly, L.T., C. Indiono, J. Mangler, and S. Rinderle-Ma, "Data Transformation and Semantic Log Purging for Process Mining." *Lecture Notes in Computer Science*, vol. 7328, pp. 238–253, 2012.

— Mans, R.S., W.M.P. van der Aalst, and R.J.B. Verwersch, *Process Mining in Healthcare: Evaluating and Exploiting Operational Healthcare Processes*. Heidelberg: Springer-Verlag, 2015.

— Mans, R.S., W.M.P. van der Aalst, R.J.B. Vanwersch, A.J. Moleman, "Process Mining in Healthcare: Data Challenges When Answering Frequently Posed Questions," *Lecture Notes in Computer Science*, vol. 7738, pp. 140–153, 2013.

— Mueller-Wickop, N. and M. Schultz, "ERP Event Log Preprocessing: Timestamps vs. Accounting Logic." *Lecture Notes in Computer Science*, vol. 7939, pp. 105–119, 2013.

— Nammakhunt, A., W. Romsaiyud, P. Porouhan, and W. Premchaiswadi, "Process mining: Converting data from MS-Access Database to MXML Format." *Tenth International Conference on ICT and Knowledge Engineering*, pp. 205–212, 2012.

— Redlich, D., W.Gilani, T. Molka, M. Ddrobek, A. Rashid, and G. Blair, "Introducing a Framework for Scalable Dynamic Process Discovery." *Lecture Notes in Business Information Processing*, vol. 174, pp. 151–166, 2014.

— Sahlabadi, M., R.C. Muniyandi, and Z. Shukur, "Detecting Abnormal Behavior in Social Network Websites by using a Process Mining Technique." *Journal of Computer Science*, vol. 10, no. 3, pp. 393–420, 2014.

— Seeber, I., R. Maier, and B. Weber, "CoPrA: A Process Analysis Technique to Investigate Collaboration in Groups." *45th Hawaii International Conference on System Science (ICSS)*, pp. 362–372, 2012.

— Sellami, R, W. Gaaloul, W. S. and Moalla, "An Ontology for Workflow Organizational Model Mining." *IEEE 21st International Workshop for Collaborative Enterprises (WETICE)*, pp. 199–204, 2013.

— Singh, L. and G. Chetty, “A Comparative Study of MRI Data using Various Machine Learning and Pattern Recognition Algorithms to Detect Brain Abnormalities.” *Proceedings of Data Mining and Analytics*, pp. 157–165, 2012.

— Suriadi, S., C. Ouyang, W.M.P. van der Aalst, and A.H.M ter Hofstede, “Root Cause Analysis with Enriched Process Logs.” *Lecture Notes in Business Information Processing*, vol. 132, pp. 174–186, 2012.

— van den Broucke, S.K.L.M., J. De Weerdt, J. Vanthienen, and B. Baesens, “A comprehensive benchmarking framework (CoBeFra) for conformance analysis between procedural process models and event logs in ProM,” *IEEE Symposium on Computational Intelligence and Data Mining*, pp. 254–261, 2013.

— van der Aalst, W.M.P., *Process Mining: Discovery, Conformance and Enhancement*. Heidelberg: Springer-Verlag, 2011.

— van der Aalst, W.M.P., et al., “Process Mining Manifesto.” in *Lecture Notes in Business Information Processing*, vol. 99, pp. 169–194, 2012.

— van der Aalst, W.M.P. and B.F. van Dongen, “Discovering Petri Nets from Event Logs.” *Lecture Notes in Computer Science*, vol. 7480, pp. 372–422, 2013.

— van der Werf, J.M.E.M. and H.M.W. Verbeek, “Online Compliance Monitoring of Service Landscapes.” *Lecture Notes in Business Information Processing*, vol. 202, pp. 89–95, 2015.

— van Dongen, B.F. and S. Shabani, “Relational XES: Data Management for process mining,” *27th International Conference on Advanced Information Systems Engineering*, pp. 169–176, 2015.

— Vasilecas, O., T. Svickas, and E. Lebedys, “Directed Acyclic Graph Extraction from Event Logs.” *Communications in Computer and Information Science*, vol. 465, pp. 172–181, 2014.

— Vera-Baquero, A., R. Colomo-Palacios, O. Molloy, and M Elbattah, “Business process improvement by means of Big Data based Decision Support Systems: a case study on Call Centers.” *International Journal of Information Systems and Project Management*, vol. 3, no. 1, pp. 5–26, 2015.

— Verbeek, H.M.W., J.C.A.M. Buijs, B.F. van Dongen, and W.M.P. van der Aalst, “XES, XESame, and ProM 6.” *Lecture Notes in Business Information Processing*, vol. 72, pp. 60–75, 2011 [B30].

— Wakup, C. and J. Desel, “Analyzing a TCP/IP-Protocol with Process Mining Techniques.” *Lecture Notes in Business Information Processing*, vol. 202, pp. 353–364, 2015.

— Weber, P., “A Framework for the Analysis and Comparison of Process Mining Algorithms.” PhD thesis, University of Birmingham, 2013.

— Westergaard, M. and van Dongen, B.F., “KeyValuesets: Event Logs Revisited,” BPM Center Report BPM-13-25, 2013. Available at: http://bpmcenter.org/wp-content/uploads/reports/2013/BPM-13-25.pdf

— Wynn, M.T., W.Z. Low, A.H.M. ter Hofstede, and W. Nauta, “A Framework for Cost-Aware Process Management: Cost Reporting and Cost Prediction.” *Journal of Universal Computer Science*, vol. 20, no. 3, pp. 406–430, 2014.

— zur Muehlen, M. and K.D. Swenson, “BPAF: A Standard for the Interchange of Process Analytics Data, *Lecture Notes in Business Information Processing*, vol. 66, pp. 170–181, 2011 [B34].

Annex D

(normative)

XES Schema definition (XSD)

This XES XML Schema definition is based on the types defined in XML Schema Part 2: Datatypes, Second Edition and the structure types defined in XML Schema Part 1: Structures, Second Edition.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified">

  <xs:element name="log" type="LogType"/>

  <!-- Attributables -->
  <xs:complexType name="AttributableType">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="string" minOccurs="0" maxOccurs="unbounded"
                  type="AttributeStringType" />
      <xs:element name="date" minOccurs="0" maxOccurs="unbounded"
                  type="AttributeDateType" />
      <xs:element name="int" minOccurs="0" maxOccurs="unbounded"
                  type="AttributeIntType" />
      <xs:element name="float" minOccurs="0" maxOccurs="unbounded"
                  type="AttributeFloatType" />
      <xs:element name="boolean" minOccurs="0" maxOccurs="unbounded"
                  type="AttributeBooleanType" />
      <xs:element name="id" minOccurs="0" maxOccurs="unbounded"
                  type="AttributeIDType" />
      <xs:element name="list" minOccurs="0" maxOccurs="unbounded"
                  type="AttributeListType" />
    </xs:choice>
  </xs:complexType>

  <!-- String attribute -->
  <xs:complexType name="AttributeStringType">
    <xs:complexContent>
      <xs:extension base="AttributeType">
        <xs:attribute name="value" use="required" type="xs:string" />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- Date attribute -->
  <xs:complexType name="AttributeDateType">
    <xs:complexContent>
      <xs:extension base="AttributeType">
        <xs:attribute name="value" use="required" type="xs:dateTime" />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
```

```
<!-- Integer attribute -->
<xs:complexType name="AttributeIntType">
  <xs:complexContent>
    <xs:extension base="AttributeType">
      <xs:attribute name="value" use="required" type="xs:long" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- Floating-point attribute -->
<xs:complexType name="AttributeFloatType">
  <xs:complexContent>
    <xs:extension base="AttributeType">
      <xs:attribute name="value" use="required" type="xs:double" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- Boolean attribute -->
<xs:complexType name="AttributeBooleanType">
  <xs:complexContent>
    <xs:extension base="AttributeType">
      <xs:attribute name="value" use="required" type="xs:boolean" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- ID attribute -->
<xs:complexType name="AttributeIDType">
  <xs:complexContent>
    <xs:extension base="AttributeType">
      <xs:attribute name="value" use="required" type="xs:string" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- List attribute -->
<xs:complexType name="AttributeListType">
  <xs:complexContent>
    <xs:extension base="AttributeType">
      <xs:sequence>
        <xs:element name="values" minOccurs="1" maxOccurs="1"
                    type="AttributeType" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- Extension definition -->
<xs:complexType name="ExtensionType">
  <xs:attribute name="name" use="required" type="xs:NCName" />
  <xs:attribute name="prefix" use="required" type="xs:NCName" />
  <xs:attribute name="uri" use="required" type="xs:anyURI" />
</xs:complexType>
```

```
<!-- Globals definition -->
<xs:complexType name="GlobalsType">
  <xs:complexContent>
    <xs:extension base="AttributableType">
      <xs:attribute name="scope" type="xs:NCName" use="required" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- Classifier definition -->
<xs:complexType name="ClassifierType">
  <xs:attribute name="name" type="xs:NCName" use="required" />
  <xs:attribute name="scope" type="xs:NCName" use="required" />
  <xs:attribute name="keys" type="xs:token" use="required" />
</xs:complexType>

<!-- Attribute -->
<xs:complexType name="AttributeType">
  <xs:complexContent>
    <xs:extension base="AttributableType">
      <xs:attribute name="key" use="required" type="xs:Name" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- Elements may contain attributes -->
<xs:complexType name="ComponentType">
  <xs:complexContent>
    <xs:extension base="AttributableType" />
  </xs:complexContent>
</xs:complexType>

<!-- Logs are elements that may contain traces -->
<xs:complexType name="LogType">
  <xs:complexContent>
    <xs:extension base="ComponentType">
      <xs:sequence>
        <xs:element name="extension" minOccurs="0"
                    maxOccurs="unbounded" type="ExtensionType" />
        <xs:element name="global" minOccurs="0"
                    maxOccurs="unbounded" type="GlobalsType" />
        <xs:element name="classifier" minOccurs="0"
                    maxOccurs="unbounded" type="ClassifierType" />
        <xs:element name="trace" minOccurs="0" maxOccurs="unbounded"
                    type="TraceType" />
        <xs:element name="event" minOccurs="0" maxOccurs="unbounded"
                    type="EventType" />
      </xs:sequence>
      <xs:attribute name="xes.version" type="xs:decimal"
                    use="required" />
      <xs:attribute name="xes.features" type="xs:token" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

```
  <!-- Traces are elements that may contain events -->
  <xs:complexType name="TraceType">
    <xs:complexContent>
      <xs:extension base="ComponentType">
        <xs:sequence>
          <xs:element name="event" minOccurs="0" maxOccurs="unbounded"
                      type="EventType"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!-- Events are elements -->
  <xs:complexType name="EventType">
    <xs:complexContent>
      <xs:extension base="ComponentType">
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

</xs:schema>
```

Annex E

(normative)

XESEXT Schema definition (XSD)

This XESEXT XML Schema definition is based on the types defined in XML Schema Part 2: Datatypes, Second Edition and the structure types defined in XML Schema Part 1: Structures, Second Edition.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           elementFormDefault="qualified">

  <xs:element name="xesextension">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="log" type="AttributableType" minOccurs="0" />
        <xs:element name="trace" type="AttributableType"
                    minOccurs="0" />
        <xs:element name="event" type="AttributableType"
                    minOccurs="0" />
        <xs:element name="meta" type="AttributableType"
                    minOccurs="0" />
      </xs:sequence>
      <xs:attribute name="name" type="xs:NCName" use="required" />
      <xs:attribute name="prefix" type="xs:NCName" use="required" />
      <xs:attribute name="uri" type="xs:anyURI" use="required" />
    </xs:complexType>
  </xs:element>

  <!-- Attributes -->
  <xs:complexType name="AttributableType">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="string" type="AttributeType" />
      <xs:element name="date" type="AttributeType" />
      <xs:element name="int" type="AttributeType" />
      <xs:element name="float" type="AttributeType" />
      <xs:element name="boolean" type="AttributeType" />
      <xs:element name="id" type="AttributeType" />
      <xs:element name="list" type="AttributeType" />
    </xs:choice>
  </xs:complexType>

  <!-- Attribute -->
  <xs:complexType name="AttributeType">
    <xs:sequence>
      <xs:element name="alias" type="AliasType"
                  minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:attribute name="key" type="xs:Name" use="required" />
  </xs:complexType>

  <!--  Alias definition, defining a mapping alias for an attribute -->
  <xs:complexType name="AliasType">
    <xs:attribute name="mapping" type="xs:NCName" use="required" />
    <xs:attribute name="name" type="xs:string" use="required" />
  </xs:complexType>

</xs:schema>
```

Annex F

(informative)

XESEXT Example

We have chosen the Semantic Extension (see 7.5) to exemplify the XESEXT format for XES extensions. This extension defines, on each level of abstraction (log, trace, event, and meta), the same string-based attribute modelReference. Attributes can be defined on all four levels of abstraction, similar to attribute declarations in XES (while omitting the value attribute). For every defined attribute, the XESEXT document may feature an arbitrary number of alias mappings as child elements. These mappings define a human-readable alias for the attribute within a given namespace (typically a country code, used for localization).

```
<?xml version="1.0" encoding="UTF-8" ?>
<xesextension name="Semantic"
              prefix="semantic"
              uri="http://www.xes-standard.org/semantic.xesext">

  <log>
    <string key="modelReference">
      <alias mapping="EN" name="Ontology Model Reference" >
      <alias mapping="DE" name="Ontologie-Modellreferenz" />
      <alias mapping="FR" name="Référence au Modèle Ontologique" />
      <alias mapping="ES" name="Referencia de Modelo Ontológico" />
      <alias mapping="PT" name="Referência de Modelo Ontológico" />
    </string>
  </log>

  <trace>
    <string key="modelReference">
      <alias mapping="EN" name="Ontology Model Reference" />
      <alias mapping="DE" name="Ontologie-Modellreferenz" />
      <alias mapping="FR" name="Référence au Modèle Ontologique" />
      <alias mapping="ES" name="Referencia de Modelo Ontológico" />
      <alias mapping="PT" name="Referência de Modelo Ontológico" />
    </string>
  </trace>

  <event>
    <string key="modelReference">
      <alias mapping="EN" name="Ontology Model Reference" />
      <alias mapping="DE" name="Ontologie-Modellreferenz" />
      <alias mapping="FR" name="Référence au Modèle Ontologique" />
      <alias mapping="ES" name="Referencia de Modelo Ontológico" />
      <alias mapping="PT" name="Referência de Modelo Ontológico" />
    </string>
  </event>

  <meta>
    <string key="modelReference">
      <alias mapping="EN" name="Ontology Model Reference" />
      <alias mapping="DE" name="Ontologie-Modellreferenz" />
      <alias mapping="FR" name="Référence au Modèle Ontologique" />
      <alias mapping="ES" name="Referencia de Modelo Ontológico" />
      <alias mapping="PT" name="Referência de Modelo Ontológico" />
    </string>
  </meta>

</xesextension>
```

Annex G

(informative)

Bibliography

Bibliographical references are resources that provide additional or helpful material but do not need to be understood or used to implement this standard. Reference to these resources is made for informational use only. All links to sources were verified as of 24 October 2016.

[B1] Agrawal, R., D. Gunopulos, and F. Leymann, "Mining Process Models from Workflow Logs," Lecture Notes in Computer Science, vol. 1377, pp. 469–483, November 1998.http://dx.doi.org/10.1007/BFb0101003[11]

[B2] AProMore website, http://apromore.org/.

[B3] Celonis website, http://www.celonis.de/en/.

[B4] CoPrA Tool website: http://www.copra-tool.eu/.

[B5] Disco website: http://fluxicon.com/disco.

[B6] "Flipflop: A Test and Flip Net Synthesis Tool for Maintenance and Surgical Process Mining," Available at: http://tinyurl.com/oql6f3y.

[B7] Günther, C.W., "Process Mining in Flexible Environments." PhD thesis, Eindhoven University of Technology, 2009.

[B8] IEEE CIS Task Force on Process Mining website: http://www.win.tue.nl/ieeetfpm.

[B9] IEEE Computational Intelligence Society, "The eXtensible Event Stream (XES) standard," XES Webinar, IEEE CIS Video Collection: http://cis.ieee.org/component/content/article/5-education-content/502-the-extensible-event-stream-xes-standard.html.

[B10] JIRAVIEW website: https://github.com/godatadriven/jiraview.

[B11] Lyytinen, K., L. Mathiassen, J. Ropponen, and A. Datta, "Automating the Discovery of As-Is Business Process Models: Probabilistic and Algorithmic Approaches," Information Systems Research, vol. 9, no. 3, pp. 275–301, March 1998, http://dx.doi.org/10.1287/isre.9.3.275.[12]

[B12] minit website: http://minitlabs.com.

[B13] OpenXES website: http://www.openxes.org.

[B14] ProM website: http://www.promtools.org/prom6.

[B15] Rai, S., C.B. Duke, V. Lowe, C. Quan-Trotter, and T. Scheermesser, "LDP Lean Document Production—O.R.-Enhanced Productivity Improvements for the Printing Industry," Interfaces, vol. 39, no. 1, pp. 69–90, 2009, http://dx.doi.org/10.1287/inte.1080.0413.

[B16] RapidProM website: http://rapidprom.org/.

[B17] Rialto website: http://www.exeura.eu/en/products/rialto.

[11]Springer publications are available at http://www.link.springer.com.
[12]ACM publications are available at: http://dl.acm.org.

[B18] Rozinat, A. and W.M.P. van der Aalst, "Conformance checking of processes based on monitoring real behavior," Information Systems, vol. 33, no. 1, pp. 64–95, March 2008.http://dx.doi.org/10.1016/j.is.2007.07.001[13]

[B19] The Ruby Toolbox website: https://www.ruby-toolbox.com/projects/ruby-xes.

[B20] Steeman, W., BPI Challenge 2013 incidents data set: http://dx.doi.org/10.4121/uuid:500573e6-accc-4b0c-9576-aa5468b10cee.

[B21] Universitat Politècnica de Catalunya, "PMLAB", https://github.com/josepcarmona/PMLAB.

[B22] van den Broucke, S., J. De Weerdt, J. Vanthienen, and B. Baesens, "CoBeFra: A Comprehensive Benchmarking Framework for Conformance Checking," Available at: http://processmining.be/cobefra/

[B23] van der Aalst, W.M.P., B.F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A.J.M.M. Weijters, "Workflow mining: A survey of issues and approaches," Data & Knowledge Engineering, vol. 47, no. 2, pp. 237–267, November 2003, http://dx.doi.org/10.1016/S0169-023X(03)00066-1.

[B24] van der Aalst, W.M.P., H.A. Reijers, A.J.M.M. Weijters, B. F. van Dongen, A.K. Alves de Medeiros, M. Song, and H.M.W. Verbeek, "Business process mining: An industrial application," Information Systems, vol. 32, no. 5, pp. 713–732, July 2007, http://dx.doi.org/10.1016/j.is.2006.05.003.

[B25] van der Aalst, W.M.P., A.J.M.M. Weijters, and L. Maruster, "Workflow Mining: Discovering Process Models from Event Logs," IEEE Transactions on Knowledge and Data Engineering, vol. 16, no. 9, pp. 1128–1142, September 2004, http://dx.doi.org/10.1109/TKDE.2004.47.[14,15]

[B26] van Dongen, B.F., BPI Challenge 2012 data set: http://dx.doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f.

[B27] van Dongen, B.F., BPI Challenge 2015 data set: http://dx.doi.org/10.4121/uuid:31a308ef-c844-48da-948c-305d167a0ec1.

[B28] van Dongen, B.F., Real-life event logs—Hospital log data set: http://dx.doi.org/10.4121/uuid:d9769f3d-0ab0-4fb8-803b-0d1120ffcf54.

[B29] van Dongen, B.F. and W.M.P. van der Aalst, "A Meta Model for Process Mining Data." CAiSE 2005 Workshops, Vol. 2. EMOI-INTEROP Workshop, 2005, pp. 309–320.

[B30] Verbeek, H.M.W., J.C.A.M. Buijs, B.F. van Dongen, and W.M.P. van der Aalst, "XES, XESame, and ProM 6," Lecture Notes in Business Information Processing, vol. 72, pp. 60–75, 2011, http://dx.doi.org/10.1007/978-3-642-17722-4_5.

[B31] XES 1.3 website: https://pypi.python.org/pypi/xes/.

[B32] XES Standard website: http://www.xes-standard.org.

[B33] YAWL website: http://yawlfoundation.org/.

[B34] zur Muehlen, M. and K.D. Swenson, "BPAF: A Standard for the Interchange of Process Analytics Data," Lecture Notes in Business Information Processing, vol. 66, pp. 170–181, 2011, http://dx.doi.org/10.1007/978-3-642-20511-8_15.

[13]Elsevier publications are available at http://www.sciencedirect.com.
[14]The IEEE standards or products referred to in this clause are trademarks of The Institute of Electrical and Electronics Engineers, Inc.
[15]IEEE publications are available from The Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08854, USA (http://ieeexplore.ieee.org/).