SNPWG15 –12

15th SNPWG MEETING
Helsinki, Finland, 12-16 November, 2012

Inclusion of a Temporal Model within S-100

Submitted by UK/Jeppesen

Introduction

Some time ago the UK raised concerns about the ability of the date types in S-100 to support the periodic date structure used in S-57. Following this Jeppesen submitted a paper to the Data Classification and Encoding Guide (DCEG) sub group prior to TSMAD 24 which formalised this discussion and presented two options;

1. Amend S-100 and S-101 to permit use of earlier editions of ISO 8601 and continue to use the S-57 formats; or,
2. Confirm that S-100 representations of dates and times must conform to ISO 8601:2004, and revise the dates/times model of S-101 to use a different representation for recurring intervals. Defining new attributes for "number of recurrences" and "duration" as in ISO 8601:2004 should work, but will make the model a little more complex.

This paper proposes a way forward based on option 2, a draft is included at annexe A. This ensures alignment with modern standards; it requires that a short profile of ISO 19108 be added to S-100 further detailing how date and time information should be used within S-100 Product Specifications. This profile will ensure consistent and clear use of temporal information between S-100 based products and support improved information interchange.

Conclusion

S-100 currently does not support truncated date time types. The proposed temporal model will allow S-100 to support these types and S-10x products use these as required within their data models. This model has not yet been considered by TSMAD but as a stakeholder SNPWG's views are welcomed at this time as input to this work.

Action required of SNPWG

- To note the contents of the proposed temporal model
- To provide any comments on the proposed model in particular in respect of how it meets the requirements of SNPWG

Annexe A

3-6.4.4 S-100 Temporal Information

References

ISO 19108:2005
ISO 8601:2004

3-6.4.4.1 Introduction

ISO 19108 provides the concepts needed to describe the temporal characteristics of geographic information as they are abstracted from the real world. Temporal characteristics of geographic information include attributes, associations and metadata elements that take a value in the temporal domain. Time provides a fundamental element within many geographic datasets. Consistent modelling of temporal information is required to ensure consistent interaction between different S-100 products and across domains.

3-6.4.4.2 Temporal Schema

The temporal schema consists of temporal objects and temporal reference system. Temporal objects defines temporal geometric and topological objects that shall be used as values for the temporal characteristics of features and data sets. The temporal position of an object shall be specified in relation to a temporal reference system. S-100 products shall use the Gregorian Calendar and 24-hour local or Coordinated Universal Time (UTC) for information interchange as specified in ISO 8601. Where local time is used the offset from UTC must be provided see Value Types.

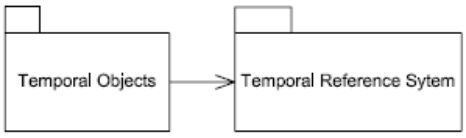

Figure 1 — Structure of the temporal schema

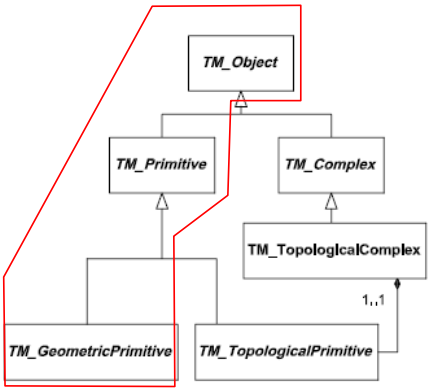

Figure 2 – Temporal primitives

3-6.4.4.3 TM Objects

S-100 constrains TM_Objects to only TM_Object, TM_Primitive and TM_Geometric Primitive. TM_Object (see Figure 2) is an abstract class that has two subclasses. TM_Primitive is an abstract class that represents a non-decomposed element of geometry or topology of time. TM_GeometricPrimitive provides information about temporal position.

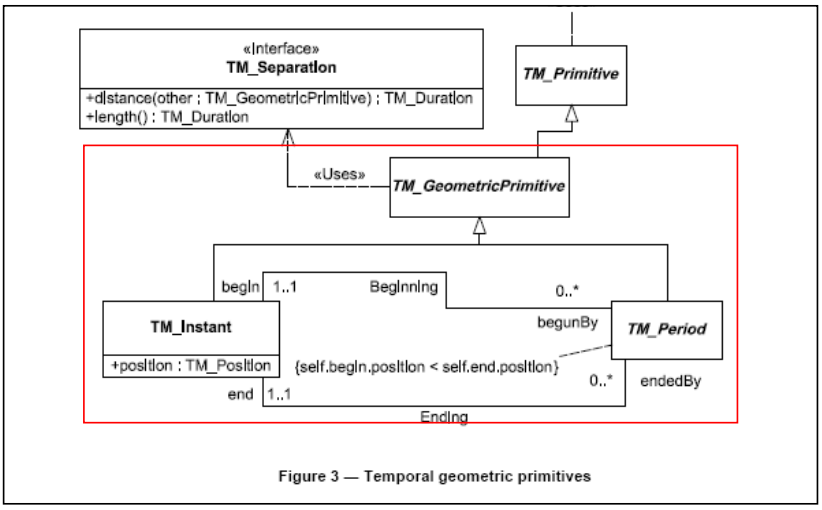

Figure 3 — Temporal geometric primitives

3-6.4.4.4 Temporal Geometric Primitives

The two geometric primitives in the temporal dimension are the instant and the period. These primitives are defined analytically in the case of time measured on an interval scale, and analogically in the case of time measured on an ordinal scale. TM_GeometricPrimitive is an abstract class with two subclasses, TM_Instant represents an instant and TM_Period represents a period (see Figure 3).

3-6.4.4.5 S100_TM_Instant

An instant is a zero-dimensional geometric primitive that represents position in time. It is equivalent to a point in space. In practice, an instant is an interval whose duration is less than the resolution of the time scale.
Attributes:

TM_Instant has one attribute

position:TM_TemporalPosition shall provide the position of this TM_Instant. An instance of TM_Instant is an identifiable object, while an instance of TM_TemporalPosition is a data value.

3-6.4.4.6 S100_TM_Period

The period is a one-dimensional geometric primitive that represents extent in time. The period is equivalent to a curve in space. Like a curve, it is an open interval bounded by beginning and end points (instants), and has length (duration). Its location in time is described by the temporal positions of the instants at which it begins and ends; its duration equals the temporal distance between those two temporal positions. Since it is impossible to measure duration on an ordinal scale, an instant cannot be distinguished from a period on this basis. In practice, the time at which a single event occurs can be considered an instant when time is measured.

a) *position:TM_TemporalPosition* shall provide the position of this TM_Instant. An instance of TM_Instant is an identifiable object, while an instance of TM_TemporalPosition is a data value. A series of consecutive events must occupy an interval of time, which is a period. The term period is commonly applied to sequences of events that have distinctive characteristics in common.

Associations:
a) *Beginning* links the TM_Period to the TM_Instant at which it starts.
b) *Ending* links the TM_Period to the TM_Instant at which it ends.

Constraints:
a) self.begin.position self.end.position states that the temporal position of the beginning of the period must be less than (i.e. earlier than) the temporal position of the end of the period.

3-6.4.4.7 S100_TM_Position

TM_Position is a union class that consists of one of the data types listed as its attributes. Date, Time, and DateTime are basic data types defined in ISO/TS 19103. They comply with ISO 8601 encoding of dates and times as character strings. These data types may be used for describing temporal positions referenced to the Gregorian calendar and UTC. TM_TemporalPosition and its subtypes shall be used for describing temporal positions referenced to other temporal reference systems. The data types defined in 5.4.4 specify numeric values for dates and times. They may be used for temporal positions referenced to any calendar or clock, including the Gregorian calendar and UTC.

3-6.4.4.8 S100_TM_TruncatedDateTimeType

S-100 extends ISO 19108 to include a specific data type for truncated date time. This ensures that partial dates can be used for recurring periods. This type is realised as a complex attributes with 4 optional elements providing the elements of the truncated dateTime.

3-6.4.4.9 S-100 Temporal Model

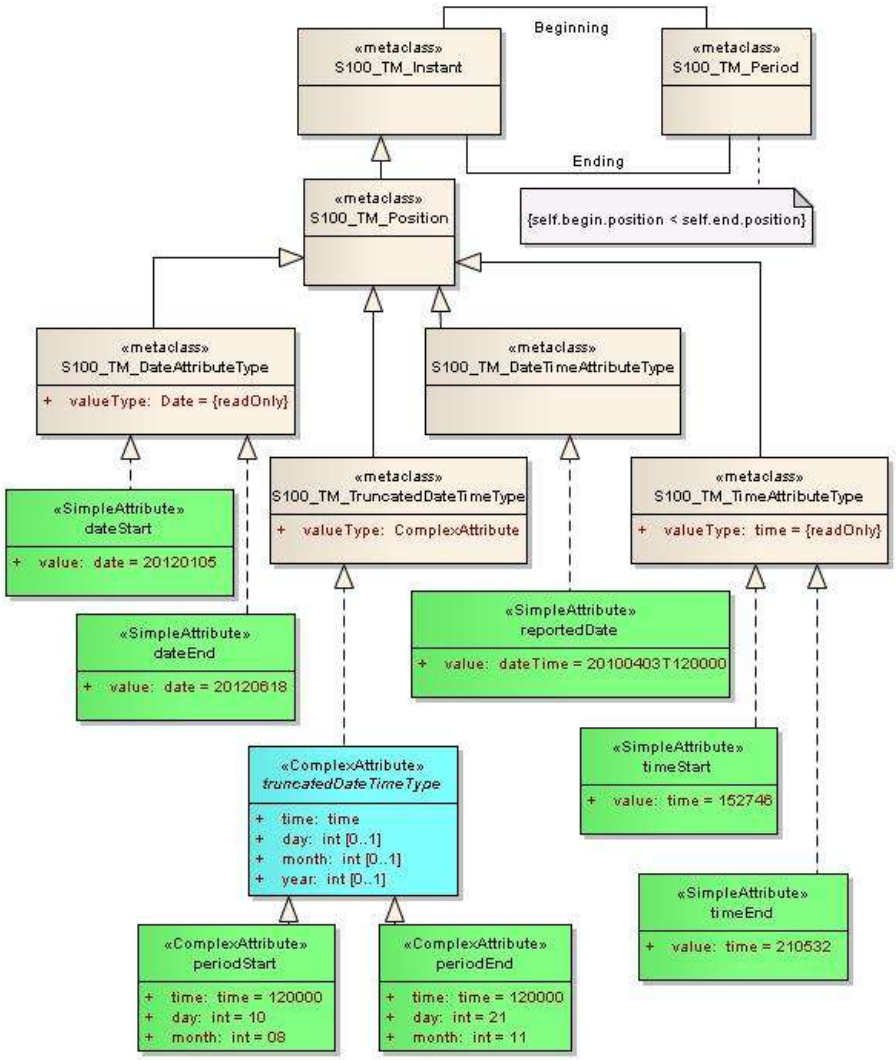

Figure 4 - S-100 Temporal Information Model – Items in green are indicative examples only.

3-6.4.4.10 Value types

Table 1. Value types

Date	A date gives values for year, month and day according to the Gregorian Calendar. Character encoding of a date is a string which shall follow the calendar date format (complete representation, basic format) for date specified by ISO 8601. EXAMPLE 19980918 (YYYYMMDD)
Time	A time is given by an hour, minute and second or fractions thereof. Character encoding of a time is a string that follows any of the time of day formats defined in ISO 8601. Product specifications shall specify which formats are allowed for their domains, and where appropriate, the decimal separator and number of digits in the decimal fraction. Time zone according to UTC is optional. EXAMPLES: a) 183059 or 183059+0100 or 183059Z (complete representation, basic format) b) 18:30:59 or 18:30:59+0100 or 18:30:59Z (complete representation, extended format) c) 1830 or 1830+0100 or 1830Z (reduced accuracy with 2 digits omitted, basic format) d) 18:30 or 18:30+0100 or 18:30Z (reduced accuracy with 2 digits omitted,, extended format) e) 18 or 18Z (reduced accuracy with 4 digits omitted, basic format – extended format is not applicable when 4 digits are omitted). f) 183059.50, 1830.7, 18.50 (decimal representations, basic format) g) 18:30:59.50, 18:30.7 (decimal representations, extended format) Note that the time designated by (c) and (d) is different from the time designated by (a) and (b) and the time designated by (e) is different from both. The complete representation of the time of 27 minutes and 46 seconds past 15 hours locally in Geneva (in winter one hour ahead of UTC), and in New York (in winter five hours behind UTC), together with the indication of the difference between the time scale of local time and UTC, are used as examples. Geneva*:* 152746+0100 New York: 152746-0500
DateTime	A DateTime is a combination of a date and a time type. Character encoding of a DateTime shall follow ISO 8601 (see above). EXAMPLE: 19850412T101530
truncatedDateTimeType	A truncatedDateType allows a partial TM Position to be given. To do this a Complex Attribute carrying the required datetime elements as individual sub attributes is used. Such a complex attribute must only use the following simple attributes; time – Time type (see above) day – integer between 1-31 month – integer between 1-12 year – integer between 0000 - 9999

3-6.4.4.11 Interpretation of interval start and end

The start and end instants of periods shall be included in the period unless a product specification specifies a different interpretation. This is based on ISO 8601:2004 § 2.1.3 which defines time interval as “the part of the time axis delimited by two instants” and provides that “A time interval comprises all instants between the two limiting instants and, unless otherwise stated, the two limiting instants themselves”.

EXAMPLES: Applying this to encoding intervals using the reduced accuracy representation or the truncatedDateTimeType, results in the interpretations in Table

2. The table also indicates how the special case of leap years can be handled.

Table 2. Examples of periods

<truncatedDateTimeType> periodStart	month=01 year, day, and time not encoded	000000 on January 1 through 240000 on the 29th day of February in leap years and the 28th day of February in non-leap years
<truncatedDateTimeType> periodEnd	month=02 year, day, and time not encoded	
<truncatedDateTimeType> periodStart	month=01 day=01 year and time not encoded	000000 on January 1 through 240000 on the 28th day of February each year
<truncatedDateTimeType> periodEnd	month=02 day=28 year and time not encoded	
<S100_TM_DateAttributeType> dateStart	dateStart=20120105	000000 on January 5, 2012 through 240000 on June 18, 2012
<S100_TM_DateAttributeType> dateEnd	dateEnd=20120618	

3-6.4.4.12 Use of specific types for truncated Date Time

Encodings may utilise specific types as supported by that encoding in order to incorporate truncated DateTime values. Where this occurs the encoding must specify the mapping between the truncatedDateTimeType values and those within the encoding specific types.

Comment [r1]: An alternative approach would be to incorporate such specific types within the model itself - discuss

Example;

An XML based encoding may define the following specific simple attribute type.

xs:gMonthDay - --12-17

The S-100 truncatedDateTime values are --(month)-(day)

Annex B. (Informative) Example: Modeling of Schedules

In the real world more complex time related scenarios exist than instants and periods. Typically a bridge may open at set times on different days of the week running to a different schedule in different seasons. A single complex attribute is used with a separate optional complex defining when the schedule applies. This allows for different schedules within different seasons. A separate complex attribute daily schedule defines the time range for each day and can be directly attached to the schedule or via a complex defining the day on which a schedule applies.

More complex schedules may require different solutions. For example, modelling holidays and special working hours can get complicated. Some considerations are:

- Some holidays occur on different days in the Gregorian calendar each year. The relevant calendrical calculations are often complicated and agencies who provide the dates of such holidays in the Gregorian calendar will probably end up supplying tables of holidays for the next N years.
- Holidays or partial working days are often declared at short notice, sometimes too short for hydrographic offices to prepare and disseminate an update.

The implication is that holidays are likely to be described in words rather than a date, and systems will not have the software logic to transform them into dates.

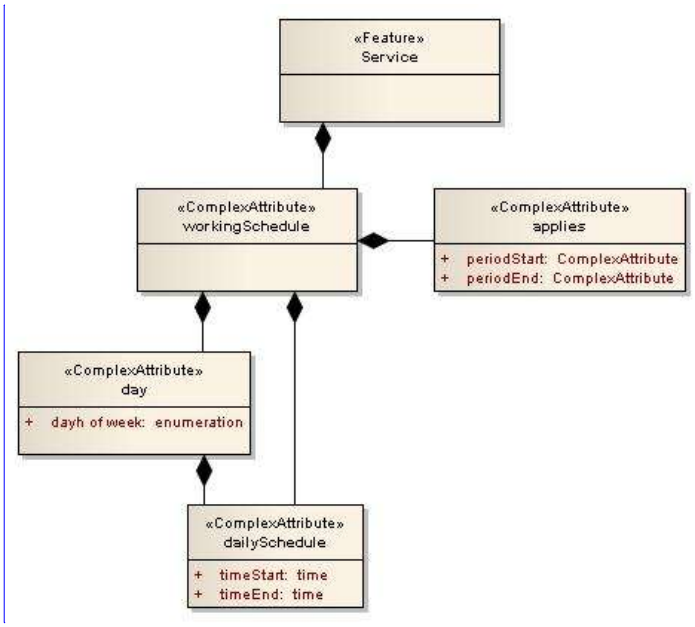

Figure 5 - Schedule example

Comment [r2]: Some names should be more precise, to avoid forcing some awkward circumlocutions and name collisions with other domains: applies -> scheduleApplies, day -> scheduleDay. Also, is workingSchedule intended to match the SNPWG feature object of the same name (SNPWG will probably want to model holidays and multiple working periods in a day).

Comment [rmm3]: A prescriptive model of schedules should be comprehensive so that all domains can use it or subsets of it. Placing anything more complicated than the basic model in XML files would be problematic because it requires applications to implement yet another "heavyweight" logic, namely, code to display the XML schedule files, in addition to code to display the basic model of schedules.

Comment [r4]: •Figure 5: How would constraints preventing duplicate or conflicting information in workingSchedule/dailySchedule and workingSchedule/day/dailySchedule be defined and encoded? Could this be done in a way which would simplify the logic and UI of tools and applications, and the work of the human hydrographer who encodes the data?

Comment [r5]: Common schedules are 24x7 and M-F, so being able to make abbreviated encodings of both would be nice. In its present form the model can abbreviate only one. Here again the point is simplifying the work of the human encoder and the logic of tool and applications.

Service	workingSchedule	applies	periodStart	month	02
			periodEnd	month	05
		day	dayOfWeek	Monday	
			dailySchedule	timeStart	080000
				timeEnd	180000
		day	dayOfWeek	Tuesday	
			dailySchedule	timeStart	070000
				timeEnd	170000
		day	dayOfWeek	Wednesday	
			dailySchedule	timeStart	090000
				timeEnd	180000
		day	dayOfWeek	Thursday	
			dailySchedule	timeStart	100000
				timeEnd	180000
	workingSchedule	applies	periodStart	month	06
			periodEnd	month	10
		dailySchedule	timeStart	090000	
			timeEnd	180000	

Figure 6 – Schedule worked example

Between February and May the service operates 8-6 Monday, 7-5 Tuesday, 9-6 Wednesday and 10-6 on Thursday. From June to October the service operates 9-6 every day.