An Ontology Design Pattern for representing Recurrent Events[⋆]

Valentina Anita Carriero[1⋆⋆], Aldo Gangemi[1,2], Andrea Giovanni Nuzzolese[1], and Valentina Presutti[1]

[1] STLab, ISTC-CNR, Rome, Italy
[2] FICLIT, University of Bologna, Bologna, Italy

Abstract. In this paper we describe an Ontology Design Pattern for modeling events that recur regularly over time and share some invariant factors, which unify them conceptually. The proposed pattern appears to be foundational, since it models the top-level domain-independent concept of recurrence, as applied to a series of events: we refer to this type of events as *recurrent event series*. The pattern relies on existing patterns, i.e. Collection, Situation, Classification, Sequence. Indeed, a recurrent event is represented as both a collection of events and a situation in which these events are contextualized and unified according to one or more properties that are peculiar to each event, and occur at regular intervals. We show how this pattern has been used in the context of ArCo, the Knowledge Graph of Italian cultural heritage, in order to model recurrent cultural events, festivals, ceremonies.

Keywords: ontology design patterns, recurrent events, collection, cultural heritage

1 Introduction

In 1917 Amedeo Modigliani, a famous Italian painter and sculptor, creates an oil painting depicting Hanka Zborowska[3], wife of his friend Léopold Zborowski. The creation of a specific painting – which is considered as a cultural property – involves the author and that cultural property, has a start date and an end date, and it occurs under particular conditions. Thus, we can assert that the creation of the Portrait of Hanka Zborowska is an event occurring only once. On the contrary, the *Art Biennale*[4] (*Biennale d'arte di Venezia*) is a contemporary visual art exhibition whose occurrences are held biennially (in odd-numbered years) since 1895, and have in common: the Biennale Foundation (*Biennale di*

[⋆] Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).
[⋆⋆] Corresponding author: valentina.carriero@istc.cnr.it
[3] https://w3id.org/arco/resource/HistoricOrArtisticProperty/1200491016
[4] https://www.labiennale.org/en/art/

Venezia) as event organizer, Venice as event place, the promotion of new contemporary art trends as mission, a time period of 2 years between two consecutive occurrences.

Given these two scenarios, it clearly appears that the second one has some peculiar properties that are not shared with the first one.

Usually, when we informally talk about the repetition of an event, we use the term *recurrent event*, as we are referring to an event (e.g. the *Art Biennale*) that occurs more than once. Actually, we are unknowingly using the term *recurrent event* to denote a series of conceptually unified events (e.g. the *Art Biennale* is a series of consecutive events that can be somehow considered as part of a uniform collection).

Facing the need to model such situations – which we refer to as recurrent event series – that involve a series of events and have some invariant and distinctive attributes in common, we realized that the concept of recurrent event is not well-represented, and cannot be captured by state-of-the-art patterns and ontologies.

In this paper we present the Recurrent Event Series Ontology Design Pattern (ODP), which models a recurrent event as a collection of events, in which the member events are in sequence and separated by a (approximately) regular time interval, and the whole collection has at least one invariant factor (e.g. the topic, the place, organizations involved, etc.), which is common to all the member events and makes the recurrent event distinctive. We assert that the proposed pattern appears to be foundational, since it models the top-level domain-independent concept of recurrence, as applied to a series of events.

The remainder of the paper is as follows: Section 2 introduces the related work, Section 3 describes the pattern and presents its formalisation by using description logics, Section 4 provides a usage example. Finally, Section 5 concludes the paper with possible future works.

2 Related work

In literature, some authors [3, 9, 8] investigate different kinds of constraints that bound a recurrent event, such as constraints related to instances of the series and constraints related to the series as a whole. Moreover, they distinguish between more or less strongly periodic events, with regard to the regularity of time intervals between the events. We follow the approach adopted by [8], where repeated events are seen as collections of events of the same type, and the notion of recurrent event is related to the idea of a coherent series of events that become part of a pattern by means of their iteration, at regular but not necessarily equal time intervals.

While in [1] the authors deal with the concept of collection, they only focus on collections of *endurants*, i.e. entities that are directly localized in space such as objects and substances, while the attributes peculiar to a collection of events, entities directly localized in time (*perdurants*), are not studied.

There exist few examples of ontologies referring to recurrent events, but in most cases they are modeled as a particular type of event, rather than as a collection of events. For instance, in DBPedia[5], recurrent event series such as annual music festivals are represented as events[6], and DBPedia ontology[7] models concepts such as `dbo:Olympics`, `dbo:Tournament` and `dbo:MusicFestival` as subclasses of `dbo:Event`. In CIDOC-CRM[8], periodic cultural events can be modeled only through the generic class `crm:E5_Event`.

We can consider an extension to the DBpedia modelling as an alternative to the one proposed here, but only if we add a mereological pattern that treats specific events as sub-events of a recurrent one. This solution may use normal interval algebra for time series of sub-events. However, a mereological approach is less clear in distinguishing typical features of recurrency. For example, a recurrent event may have no members at all, and still be a recurrent event, but in this case, a mereological whole would be empty (itd have no parts), which is both formally and intuitively impossible.

There are also other problems, e.g. the intermittent presence of the event would make it a part of a maximal continuous event organisation event, which is not the same as a collection of specifically identified events, maybe more commonsensically. In other words, if we admit that recurrency is a temporal mereology concept, we need also to admit maximal mereological wholes for any set of events that have some temporally ordered similarity, which in the considered use cases would be an overcommitment.

An approach that is closer to ours is taken by the extended module of Proton Ontology[9], where a `pext:RecurringEvent` is defined as a "recurring sequence of events" and is related to single events through the object property `pext:hasRecurringEventInstance`. The relation between the recurrent event and its frequency is expressed by the datatype property `pext:currentFrequency`. However, in this ontology a recurring event is not represented as a particular type of collection, but generally as a subclass of `proton:Happening`[10], which is defined as "something that happens", and the events that are instances of a `pext:RecurringEvent` are not represented as consecutive. Finally, the Ontology Design Pattern *PeriodicInterval*[11] provides a means to represent the period of periodic intervals, such as "every Monday", by specializing `time:Interval`[12]. Nevertheless, it is still not possible to model either factors unifying this particular type of collection or the sequence of unified events.

[5] `https://wiki.dbpedia.org/`

[6] For example, the Umbria Jazz Festival is an instance of the class `http://dbpedia.org/class/yago/Event100029378`

[7] `dbo: http://dbpedia.org/ontology/`

[8] `crm: http://www.cidoc-crm.org/cidoc-crm/`

[9] `pext: http://www.ontotext.com/proton/protonext#`

[10] `proton: https://www.ontotext.com/proton/protontop#`

[11] `http://ontologydesignpatterns.org/wiki/Submissions:PeriodicInterval`

[12] `time: https://www.w3.org/2006/time#`

3 Recurrent Event Series ODP

The presented pattern reuses as template 4 Ontology Design Patterns (ODPs): *Collection*[13], *Situation*[14], *Classification*[15], *Sequence*[16]. These patterns are extracted from DOLCE+DnS Ultra-lite[17] foundational ontology, a lighter and name-friendly of DOLCE+DnS [5], thus our ODP is also aligned to it. Moreover, all reused ODPs are annotated with OPLa ontology [6], which facilitates future reuse of this pattern.

Table 1: Competency questions answered by the Recurrent Event Series ODP.

ID	**Competency question**
CQ1	What are the events of a recurrent event series?
CQ2	Which is the time period elapsing between two events of a recurrent event series?
CQ3	When is the next event of a recurrent event series scheduled?
CQ4	What are the unifying criteria shared by all the events in a recurrent event series?
CQ5	Which is the (immediate) next event in a recurrent event series?
CQ6	Which is the (immediate) previous event in a recurrent event series?

The base namespace is `http://www.ontologydesignpatterns.org/cp/owl/recurrenteventseries.owl#` and the pattern, as submitted to the *ODP portal*, can be found at `http://ontologydesignpatterns.org/wiki/Submissions:RecurrentEventSeries`.
A diagram for the pattern is given in Figure 1, and the Competency Questions (CQs) our pattern can answer are included in Table 1.

The aim of this paper is twofold: (i) introducing a general pattern and (ii) its usage example. The general pattern provides an abstract reusable template for modelling recurrent event series that does not specify any semantics for the concepts of event and time period, which are not native to this pattern. The usage example implements the pattern by reusing the semantics associated with: `d0:Eventuality`, from the foundational ontology DOLCE-Zero[18], as a top-level concept for any event, situation or activity, and the conceptualisation of time periods as defined by the *Time Period* Content ODP[19], which is an additional original contribution of this paper. Classes and properties of the pattern are explained below.

[13] http://ontologydesignpatterns.org/wiki/Submissions:Collection
[14] http://ontologydesignpatterns.org/wiki/Submissions:Situation
[15] http://ontologydesignpatterns.org/wiki/Submissions:Classification
[16] http://ontologydesignpatterns.org/wiki/Submissions:Sequence
[17] dul: http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
[18] d0: http://www.ontologydesignpatterns.org/ont/d0.owl#
[19] http://ontologydesignpatterns.org/wiki/Submissions:TimePeriod

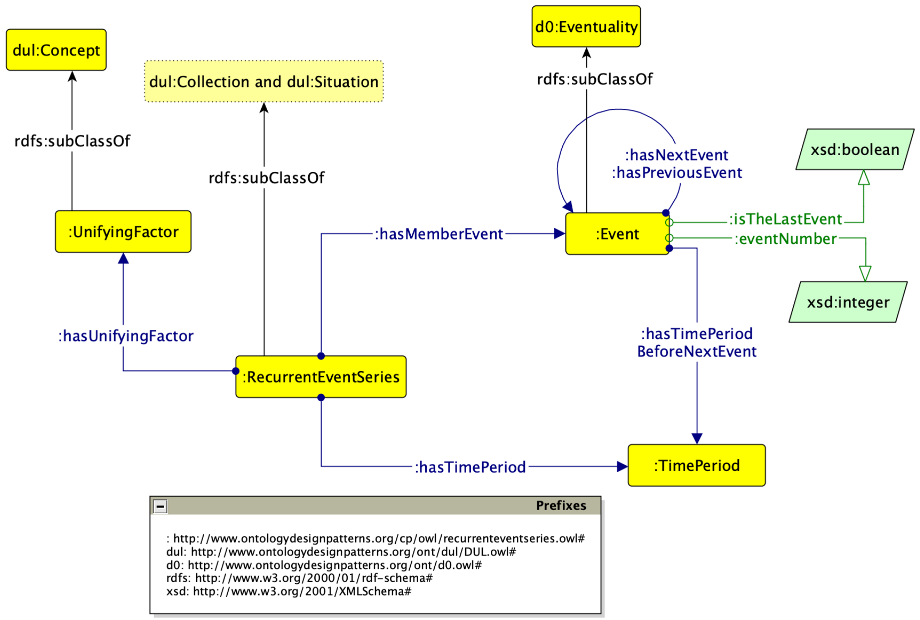
dul:Concept
dul:Collection and dul:Situation
d0:Eventuality
rdfs:subClassOf
rdfs:subClassOf
rdfs:subClassOf
:hasNextEvent
:hasPreviousEvent
xsd:boolean
:UnifyingFactor
:hasMemberEvent
:Event
:isTheLastEvent
:eventNumber
xsd:integer
:hasUnifyingFactor
:hasTimePeriod BeforeNextEvent
:RecurrentEventSeries
:hasTimePeriod
:TimePeriod
Prefixes
: http://www.ontologydesignpatterns.org/cp/owl/recurrenteventseries.owl#
dul: http://www.ontologydesignpatterns.org/ont/dul/DUL.owl#
d0: http://www.ontologydesignpatterns.org/ont/d0.owl#
rdfs: http://www.w3.org/2000/01/rdf-schema#
xsd: http://www.w3.org/2001/XMLSchema#

Fig. 1: Pattern for Recurrent Event Series, in Graffoo notation [4]. A recurrent event series is related to: its member events, which are in sequence; one ore more unifying factors; the time period between two consecutive events.

RecurrentEventSeries. Recurrent events are represented as individuals of the class `:RecurrentEventSeries`, which is modeled as an intersection of the classes `dul:Collection` and `dul:Situation`. Indeed, a recurrent event is seen as a collection, since it contains entities that share one or more common properties and are conceptually unified. These entities are member of the collection, and are consecutive events, so that each event may have either a previous event, or a next event, or both, unless the recurrent event, planned to be a series of at least two events, is interrupted after the first event, or is never instantiated. At the same time, a recurrent event is a situation, intended as a relational context in which the contextualized things are based on a frame: a recurrent event is similar to a plan that defines how the things involved in that plan (i.e. the specific events) shall be carried out, e.g. where the events shall be located, in which time of the year, etc.

$$\begin{aligned}
\texttt{RecurrentEventSeries} &\sqsubseteq \textit{dul:}\texttt{Collection} \sqcap \textit{dul:}\texttt{Situation} \\
\texttt{RecurrentEventSeries} &\equiv \exists\texttt{hasMemberEvent}. \\
&\quad (\geqslant 0\texttt{hasNextEvent}.\texttt{Event} \sqcup \\
&\quad \geqslant 0\texttt{hasPreviousEvent}.\texttt{Event})
\end{aligned}$$

hasMemberEvent. A recurrent event is a particular type of collection where all the members are events: entities flowing in time, related to space and time and involving some objects. The relation between a collection of recurring events (a `:RecurrentEventSeries`) and the events that are members of the collection is represented by the object property `:hasMemberEvent`. Its inverse property is `:isEventMemberOf`. The concept of event is not native to our pattern, thus we create a generic class `:Event` that is declared as a subclass of `d0:Eventuality`.

$$\begin{aligned}\texttt{RecurrentEventSeries} &\sqsubseteq \exists\texttt{hasMemberEvent.Event}\\ \texttt{Event} &\sqsubseteq \textit{d0:}\texttt{Eventuality}\end{aligned}$$

UnifyingFactor. A recurrent event is a collection whose members are *unified* by a unity criterion [1], i.e. organized according to one (or more if it evolves through time) recognizable pattern shared by all members. Thus, it is perceived as a unitary collection of events when it has at least one element or property that occurs in each event member, making all the events attributable to a homogeneous collection. For instance, all the events member of a recurrent event series can be located in the same place, be on the same topic, plan certain activities, involve the same audience, etc. These unifying factors may have some degree of flexibility, in order to envision properties that usually occur in all the events, but can undergo change under particular conditions. The `:RecurrentEventSeries` is related to these invariant `:UnifyingFactor`s by means of the object property `:hasUnifyingFactor`. A `:UnifyingFactor` is modelled as a subclass of `dul:Concept`, since it is a social object created for collecting existing entities, i.e. events.

$$\begin{aligned}\texttt{RecurrentEventSeries} &\sqsubseteq \exists\texttt{hasUnifyingFactor.UnifyingFactor}\\ \texttt{UnifyingFactor} &\sqsubseteq \textit{dul:}\,\texttt{Concept}\end{aligned}$$

hasNextEvent, hasPreviousEvent. The events which are members of a recurrent event series are in sequence: it is possible to represent which is the previous and/or the next event of a particular member event by means of the object properties `:hasPreviousEvent` and `:hasNextEvent`, which are defined as inverse properties. These properties are further specialised into `:hasImmediatePrevious Event` and `:hasImmediateNextEvent`, respectively, in order to associate events to their immediate next and previous events.
In the context of this pattern, by defining these properties we want to express a relation between events that are members of the same collection, that is the same recurrent event series, rather than relate an event to any other previous or next event. However, this intended restriction can not be formally expressed efficiently and in a straightforward manner with OWL 2 DL primitives [7]. Instead, it is possible to check the local integrity of an instantiation of a `:RecurrentEventSeries` by using inference rules.

For example, let us consider the following SPARQL query. An event – member of a recurrent event series – can be related by the property `:hasNextEvent` to another event – also member of a recurrent event series – only if both events are member of the same recurrent event series. Therefore, a triple asserting a local inconsistency between the two recurrent event series will be generated, if an `owl:sameAs` relation between the two recurrent event series does not exist. Similar SPARQL queries can be executed for the related constraints (e.g. events connected by `:hasPreviousEvent` object property). These queries can be embedded in the RDF model via SPIN[20] or SHACL[21].

```
PREFIX recurrentevent: <http://www.ontologydesignpatterns.org/cp/
                                    owl/recurrenteventseries.owl#>
CONSTRUCT {?re1 recurrentevent:isLocallyInconsistentWith ?re2}
WHERE {
       ?re1 recurrentevent:hasMemberEvent ?event1 .
       ?re2 recurrentevent:hasMemberEvent ?event2 .
       ?event1 recurrentevent:hasNextEvent ?event2 .
       filter not exists {?re1 owl:sameAs ?re2}
       filter (?re1 != ?re2)
       filter (?event1 != ?event2)
      }
```

The datatype property `:isTheLastEvent`, whose range is `xsd:boolean`, allows us to express whether an event is the last event of the series or not, while through the datatype property `:eventNumber` it is possible to represent the number of the event within the series (e.g. "1" for the first event).

TimePeriod. A collection of events can be defined recurrent if there is a regularity in the intervals between consecutive events: it has to be specified a temporal distance, even approximated, that elapses between one member event and the next and the previous ones. While considering necessary to explicitly express this time period, we leave the user of this ODP free to model this concept in different ways. Hence, we create a generic class `:TimePeriod`. Nevertheless, we propose a possible solution for implementing the pattern, by reusing the class `tp:TimePeriod` from the *TimePeriod* ODP[22] in our usage example (see Section 4). A `tp:TimePeriod` is related to a measurement unit and a measurement value. The collection of events is related to the recurrent time period with the object property `:hasTimePeriod`, which is defined as a property chain `[:hasMemberEvent` $\circ$ `:hasTimePeriodBeforeNextEvent]`. The object property `:hasTimePeriodBeforeNextEvent` relates an event, member of a recurrent event series, to the time period that has typically to elapse before the next event is held, which is usually approximate (e.g. yearly, monthly, etc.), and is different from the actual time between two events, which can be derived from the time intervals computed between any two members.

[20] https://spinrdf.org/
[21] https://www.w3.org/TR/shacl/
[22] tp: http://www.ontologydesignpatterns.org/cp/owl/timeperiod.owl#

$$\texttt{RecurrentEventSeries} \sqsubseteq \exists\texttt{hasTimePeriod.TimePeriod}$$
$$\texttt{hasTimePeriod} \sqsubseteq \texttt{hasMemberEvent} \circ \texttt{hasTimePeriodBeforeNextEvent}$$

TimePeriodMeasurementUnit. The `tp:TimePeriod` is associated to the unit for measuring the time interval, `tp:TimePeriodMeasurementUnit` (e.g. year, month, week), by means of the object property `tp:hasTimePeriodMeasurement Unit`. Instead, the relation with the time value is expressed by the datatype property `tp:timePeriodValue`, whose range is `xsd:integer`.

$$\texttt{TimePeriod} \sqsubseteq \exists\texttt{hasTimePeriodMeasurementUnit.TimePeriodMeasurementUnit}$$
$$\exists\texttt{timePeriodValue}.\mathit{xsd}\texttt{:integer}$$

Table 2: SPARQL queries that can be run over the Recurrent Event Series ODP. See Table 1 for the corresponding CQs.

CQ ID	**SPARQL query**
CQ1	SELECT ?event WHERE {?recurrentEvent :hasMemberEvent ?event}
CQ2	SELECT ?timePeriod WHERE {?recurrentEvent :hasTimePeriod ?timePeriod}
CQ3	SELECT ?timePeriod WHERE {?event :hasTimePeriodBeforeNextEvent ?timePeriod}
CQ4	SELECT ?unifyingFactor WHERE {?recurrentEvent :hasUnifyingFactor ?unifyingFactor}
CQ5	SELECT ?nextEvent where {?event ?p ?nextEvent . ?p rdfs:subPropertyOf* :hasNextEvent}
CQ6	SELECT ?previousEvent where {?event ?p ?previousEvent . ?p rdfs:subPropertyOf* :hasPreviousEvent}

The CQs included in Table 1 can be translated into the SPARQL queries in Table 2.

4 Usage example

We have been working on formally representing recurrent events in the context of a project named ArCo[23] (Architecture of Knowledge) [2]. During this project, we developed the Italian cultural heritage (CH) knowledge graph, consisting

[23] https://w3id.org/arco

of a network of ontologies and facts on Italian cultural properties, based on the official General Catalogue of the Italian Ministry of Cultural Heritage and Activities (MiBAC). In particular, we felt the need to represent events regularly recurring over time in two cases.

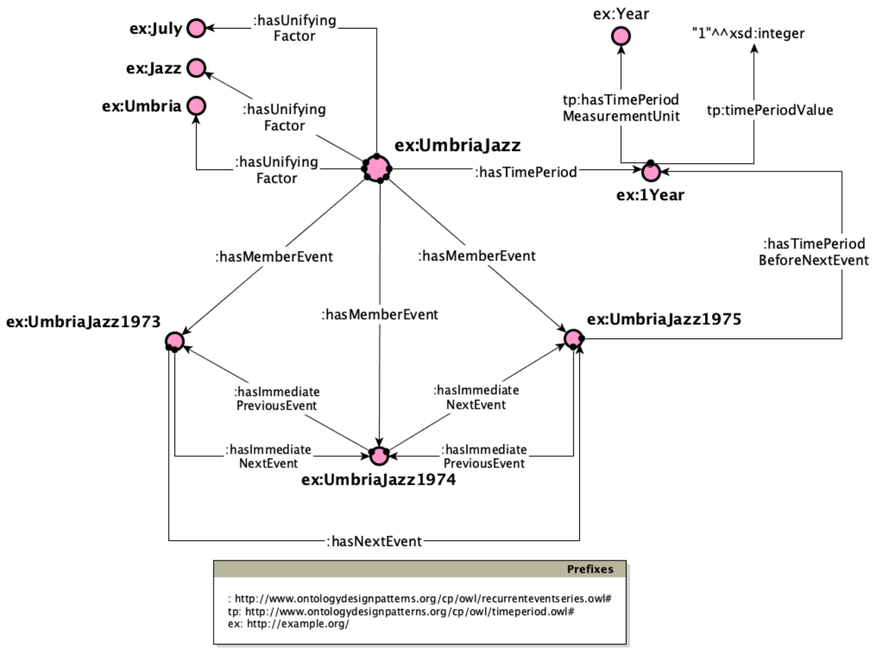

Fig. 2: Usage example in Graffoo notation. *Umbria Jazz* is a recurrent event series, and is related to: its first three consecutive events; three unifying factors; the time period between two consecutive events.

In the MiBAC catalogue records, which describe Italian cultural properties, it is possible to find information about events involving cultural properties, such as exhibitions. Cataloguers gather and record the event title, place, time, organizer(s). There is no explicit information about possible recurrent cultural events. Nonetheless, by analyzing the data, we noticed that there are many cases of event titles like "third exhibition", "tenth painting award", "first triennial exhibition", etc., which clearly refer to the same collection of events. For instance, in ArCo data, some cultural properties are involved in different editions[24] of a four-year exhibition organized by the *Quadriennale di Roma* Foundation, in Rome, out of a total of 16 exhibitions, from 1931 to 2017.

[24] E.g. this is the fourth edition, involving 3 cultural properties in the dataset: https://w3id.org/arco/resource/CulturalEvent/d13747fe737de1eda54a5250b59d91f5

The second use case is about a particular type of intangible cultural heritage: ceremonies, customs and celebrations related to the *year cycle* (e.g. Carnival, Ramadan) or to the *season cycle* (circumstances connected to e.g. popular belief, myth, science, phenomena of specific periods of the year). These kind of intangible cultural properties recur regularly, indeed the MiBAC catalogue records contain a specific tag for information about their periodicity (e.g. annual, every two years, three times a year).

Figure 2 shows a simple usage scenario. This example refers to *Umbria Jazz*, an international jazz festival held in Umbria, Italy. *Umbria Jazz* has all the attributes to be identified as a recurrent event. Indeed, it is a group of events, which can be seen as a unitarian collection on account of at least three unifying factors: it takes place in July, in the Italian region of Umbria, and has the musical genre *jazz* (and some other related genres) as a topic. Moreover, the events, which are members of this collection and have some invariant in common, recur at regular time periods: the festival is held annually, hence there is a time period of about one year between a member event and the next one (the festival does not take place on a specific date of July).

The following RDF triples, serialized as TURTLE, formally represent the example in Figure 2. The prefix `ex:` stands for a namespace used for illustrative examples (`http://example.org/`).

```
ex:UmbriaJazz a :RecurrentEventSeries;
    :hasUnifyingFactor ex:Umbria, ex:Jazz, ex:July ;
    :hasTimePeriod ex:1Year .

ex:Umbria a :UnifyingFactor .
ex:Jazz a :UnifyingFactor .
ex:July a :UnifyingFactor .

ex:1Year a tp:TimePeriod ;
    tp:hasTimePeriodMeasurementUnit ex:Year ;
    tp:timePeriodValue "1"^^xsd:integer .

:Year a tp:TimePeriodMeasurementUnit .

ex:UmbriaJazz :hasMemberEvent ex:UmbriaJazz1973 ,
                           ex:UmbriaJazz1974 , ex:UmbriaJazz1975 .

ex:UmbriaJazz1973 a :Event .
ex:UmbriaJazz1974 a :Event .
ex:UmbriaJazz1975 a :Event .

ex:UmbriaJazz1973 :hasImmediateNextEvent ex:UmbriaJazz1974 ;
    :hasNextEvent ex:UmbriaJazz1975 .

ex:UmbriaJazz1974 :hasImmediateNextEvent ex:UmbriaJazz1975 ;
    :hasImmediatePreviousEvent ex:UmbriaJazz1973 .
```

```
:UmbriaJazz1975 :hasImmediatePreviousEvent ex:UmbriaJazz1974 ;
    :hasPreviousEvent ex:UmbriaJazz1973 .

ex:UmbriaJazz1973 :hasTimePeriodBeforeNextEvent ex:1Year .
ex:UmbriaJazz1974 :hasTimePeriodBeforeNextEvent ex:1Year .
ex:UmbriaJazz1975 :hasTimePeriodBeforeNextEvent ex:1Year .
```

5 Conclusion and future work

In this paper, we proposed an Ontology Design Pattern (ODP) to represent events that recur regularly over time, in a temporal sequence, and are members of a unitary collection since they share some invariant properties. We created this pattern in response to the need of modeling cultural events, exhibitions, traditional ceremonies, festivals, etc. However, this pattern seems to be useful for representing many recurring events, which differ depending on the origin of the regular intermittence: (i) not man-made events repeating with some regularity that has a *natural origin* (e.g. the sunrise every morning or periodic migrations of animals); (ii) events where the recurrence and unifying factors are *set by humans* (e.g. a world day or a train schedule); (iii) events created by humans with a periodicity regulated by a clear purpose, as part of a *workflow* (e.g. a medical prescription defining time intervals to take a medicine or periodical supply of services).

In our future work, we aim at studying all types of recurrent events, identifying possible new features, investigating solutions for representing more or less flexible unifying factors, and at experimenting and evaluating other possible patterns for recurrent events.

References

[1] Emanuele Bottazzi et al. “From collective intentionality to intentional collectives: An ontological perspective”. In: *Cognitive Systems Research* 7.2-3 (2006), pp. 192–208.

[2] Valentina Anita Carriero et al. “ArCo: the Italian Cultural Heritage Knowledge Graph”. In: *Proceedings of ISWC 2019 - accepted for publication.* (Auckland, New Zealand). 2019.

[3] Shubha Chakravarty and Yuval Shahar. “Specification and detection of periodic patterns in clinical data”. In: *Fourth Workshop on Intelligent Data Analysis in Medicine and Pharmacology (IDAMAP-99).* Citeseer. 1999, pp. 20–31.

[4] Riccardo Falco et al. “Modelling OWL ontologies with Graffoo”. In: *European Semantic Web Conference.* Springer. 2014, pp. 320–325.

[5] Aldo Gangemi et al. “Sweetening ontologies with DOLCE”. In: *International Conference on Knowledge Engineering and Knowledge Management.* Springer. 2002, pp. 166–181.

[6] Pascal Hitzler et al. "Towards a Simple but Useful Ontology Design Pattern Representation Language". In: *Proceedings of WOP 2017*. (Vienna, Austria). 2017.

[7] Rinke Hoekstra and Joost Breuker. "Polishing diamonds in OWL 2". In: *International Conference on Knowledge Engineering and Knowledge Management*. Springer. 2008, pp. 64–73.

[8] Paolo Terenziani. "Toward a Unifying Ontology Dealing with Both User-Defined Periodicity and Temporal Constraints About Repeated Events". In: *Computational Intelligence* 18.3 (2002), pp. 336–385.

[9] Alexander Tuzhilin and James Clifford. "On Periodicity in Temporal Databases". In: *Information Systems* 20.8 (1995), pp. 619–639.