Activity Ontologies for Intelligent Calendar Applications

Agnantis Konstantinos
University of Macedonia
Dept. of Applied Informatics
Thessaloniki, Greece
kagnadis@gmail.com

Refanidis Ioannis
University of Macedonia
Dept. of Applied Informatics
Thessaloniki, Greece
yrefanid@uom.gr

ABSTRACT

Intelligent Calendar Applications (ICA) have recently emerged as a very promising target field for Artificial Intelligence (AI) techniques, since calendar applications are used daily by millions of people world-wide. ICAs built on traditional electronic calendars, by empowering them with efficient scheduling engines, being able to schedule and reschedule a user's events within its calendar. However, in order to fully exploit the dynamics of an ICA, the user has to describe his events with much more detail than with a traditional electronic calendar, in order for the scheduler to have all the necessary information, that is, attribute values, constraints and preferences, to schedule the events.

Existing XML based formats to describe events, such as iCalendar, do not provide for the information needed in order to describe a rich event, to be used by an ICA. In this paper we present three ontologies that have been designed to be used by modern ICAs, in order to exchange information about events between ICAs and event providers. The ontologies have been designed in order to describe the event, its temporal aspects, as well as the user's constraints and preferences.

Keywords

Ontology, Semantic Web, Activities, Events, Modeling, Intelligent Calendars

1. INTRODUCTION

In the recent years, Intelligent Calendar Applications (ICA) have emerged as a promising application of Artificial Intelligence planning and scheduling techniques to change the way people organize their own every-day events, actually their life ([10], [6]). Whereas the previous generation of electronic calendars, such as Microsoft Outlook or Google Calendar, focus on usability, providing many features such as periodic events, multiple calendars, shared calendars, etc., leaving the decision as of when and where to schedule each event to the user (or to the group of users, in case of meeting arrangements), ICAs focus on automating the way events are scheduled in time and space, trying to produce optimal plans in terms of a variety of criteria, that have been defined by the user.

However, in order for an ICA to produce optimal plans, a variety of details about every event included in the user's to-do list has to be provided. This is the weakest aspect of ICAs, since entering manually so many details for each event in the ICA makes their use less attractive for the majority of the potential users. So, intelligent methods need to be adopted, in order to reduce or avoid this burden, with the ontologies having to play a crucial role in them ([8], [12]).

One of the recent and very powerful ICA is SELFPLANNER ([10], [15], [16], [17]), which combines a rich model of features to describe tasks and events (referred collectively as activities in SELFPLANNER's terminology), with a very effective and efficient scheduling engine ([1], [2], [3]). Each activity in SELFPLANNER is characterized by several attributes, such as a temporal domain, consisting of a set of intervals; a minimum and a maximum duration; periodicity, without having to repeat with the same offset within each period; a set of locations where the user should be in order to accomplish the activity; its interruptibility property, with several additional features for interruptible activities; the utilization property, that is, what percentage of user's attention is consumed by the activity during its execution, and several others. Furthermore, the model allows defining binary constraints and preferences between activities, such as ordering, proximity (minimum and maximum distance) and implications constraints and preferences.

The first versions of SELFPLANNER did not allow for information exchange between the application and the users. Actually, there was no way to export the list of activities of a user, with all their details, or to import activities in some format. On the other hand, there was a rather complicated user interface, whereas the user should insert his activities manually.

This paper introduces three ontologies that have been designed to be used by ICAs. Describing activities using these ontologies should allow for exchanging activity descriptions between sites and ICAs, thus accelerating their widespread. The potential uses of such ontologies are numerous, mainly focused in cases of events that do not have a specific time and place) to occur, ranging from open hours of shops and sites, to descriptions of educational activities.

The rest of the paper is structured as following: Section 2 gives two motivational scenarios behind our work. Sections 3 through 5 present the ontologies, namely the *Activities* ontology, the *Meetings* ontology and the *CalendarPrefs* ontology, including illustrative examples. Section 6 introduced related work, while Section 7 describes the connection of our ontologies with existing ones. Finally, Section 8 concludes the paper and poses future directions.

2. MOTIVATION

We envision the following two scenarios to be realized in the next few years:

Scenario 1. John wants to schedule his weekly shopping from the local mall. He visits the web site of the mall in order to get informed about the mall's open hours. This information is provided both in pure text format, as well as in a structured way, using suitable ontologies, where all information concerning the visit to the mall is given: Open hours, location, expected duration of the visit, available means of transport, etc. John opens a file containing this information with his Intelligent Calendar Application, sets the weekend as the time window when he wants to have the visit, and finally John's ICA schedules the visit to the mall for Saturday morning, taking into account all his other commitments. Furthermore, in order to schedule this visit, other events might have been rescheduled.

Scenario 2. The Intelligent Calendar Application allows for locally storing activity descriptions. Furthermore, it is not only possible to store locally activity descriptions that have been downloaded from other web sites, but it is also possible to create and store locally new activity descriptions. John has created an activity description for performing weekend shopping, by combining individual descriptions for different malls. The combined description is actually a disjunction of individual descriptions, each one of them having each own location, open hours, etc. By introducing such a complex activity description into John's Intelligent Calendar Application, the scheduler is able to select the most convenient mall to do the weekend shopping, taking into account other user's commitments, in an attempt to minimize his travelling.

3. THE ACTIVITIES ONTOLOGY

The *Activities* ontology aims at describing any type of activity, a person would like to add to his ICA. It comprises:

- 114 classes (owl:Class)
- 207 (106+101) named individual objects (owl:NamedIndividual)
- 12 object properties (owl:ObjectProperty)
- 3 data properties (owl:DataProperty)
- 127 subclass relations (rdfs:subClassOf)
- 16 equivalent classes (owl:equivalentClass)

Among the 114 classes, 7 are the top level ones. Each one represents a notion that is important to describe an activity: The type, the physical location, the type of the location, the mean of transport, the type of participants and the temporal domain. Particularly:

Activity: This class represents the type of the activity. Any type of activity is a subclass of it.

Venue: This class represents the type of the location where the activity takes place. Any type of location is a subclass of *Venue*. There are 12 direct and 40 indirect subclasses of *Venue*, ranging from general descriptions like *PublicBuilding*, *OutDoor* and *EnterpriseHall* to more specific like *UniversityCampus*, *BeachBar* and *Office*.

GeoLocation: This class represents the physical location of a venue. For this purpose is used the property *hasLocation*, which ranges over *GeoLocation* objects, with each *GeoLocation* object having two numerical values, through the properties *hasLatitude* and *hasLongitude* respectively.

This is an example of defining a venue with its geographical coordinates (using the turtle notation):

```
##http://www.uom.gr/ai/ontologies/Activities.owl#S
uperParadise
:SuperParadise rdf:type :BeachBar ,
        owl:NamedIndividual ;
   :hasName "Super Paradise Beach Bar" ;
   :hasLocation :spLocation .
##http://www.uom.gr/ai/ontologies/Activities.owl#s
pLocation
:spLocation rdf:type :GeoLocation ,
        owl:NamedIndividual ;
   :hasLatitude "25.356245"^^xsd:double ;
   :hasLongitude "37.412437"^^xsd:double .
```

Transportation: This class represents the means of transport that can be used to go to and from the location where the activity takes place. This is an enumerated class with 13 distinct means of transport, namely (in alphabetical order): {*Bicycle*, *Boat*, *Bus*, *Car*, *Caravan*, *Metro*, *Motorcycle*, *OnFoot*, *Rollers*, *Ship*, *Taxi*, *Train*, *Tram*}.

Participant: This class represents the type of the participants in an activity. They could be one of the following values: {*Group*, *Person*, *Organization*}.

TimeInterval: This class describes a time interval through its properties *hasStartTime* and *hasEndTime*.

Finally, the *ComplexActivity* class contains two subclasses *ComplexDisjointActivity* and *ComplexUnionActivity*. These subclasses represent disjoint and union of activities, respectively.

3.1 The *Activity* class in more detail

This class aims at representing possible types of activities performed in everyday life by ordinary people. For example, indoor or outdoor activities, group or personal activities, activities where extra requirements are needed in order to be fulfilled etc. The ontology tries to distinguish all these different types and prerequisites, providing an easy and straightforward way to represent this kind of information.

Activity class has two direct subclasses: *ComplexActivity* and *SimpeActivity*. *ComplexActivity* class is used to model groups of alternative (*ComplexOneOfActivity*) or required (*ComplexAllActivity*) activities. For example, *Clean_House* activity may be composed of several other sub-activites, like *Wash_Dishes*, *Vacuum_the_floor* and *Do_the_laundry*, making it a member of the *ComplexAllActivity* class. On the other side the *Eat_Launch* activity is a member of the *ComplexOneOfActivity* class, because it can be represented by either *Cook_meal* or *Order_pizza*, as the completion of the complex activity requires the completion of just one of the alternatives.

SimpleActivity having as its single subclass, class *HumanActivity*, represents activities that are performed by human beings. All other classes are subclassses of *HumanActivity*. Direct subclasses of *HumanActivity* are the following:

- Active
- Communal
- Compulsory
- Education
- Indoor
- Leisure
- Recreation
- OutDoor
- Passive
- Solitary
- Travel
- Work

Furthermore, we have defined 92 objects that represent instances of some of the most common types of activities. Some examples are the following:

- ✓ *Going_to_shoes_store* (Shopping Activity)
- ✓ *Church_visiting* (Religious Activity)
- ✓ *Running* (Recreation Activity)
- ✓ *Attending_a_wedding* (Social Activity)
- ✓ *Taking_exams* (School Activity)
- ✓ *Painting* (Art Activity)
- ✓ *Watching_tv* (Audio Visual Activity)
- ✓ *Card_game* (Game Activity)
- ✓ *Ironing* (Housekeeping Activity)
- ✓ *Playing_basketball* (Recreation Activity)
- ✓ *Sunbathing* (Beach Activity)
- ✓ *Visiting_theater* (Cultural Activity)
- ✓ *Underwater_photography* (Water Activity)

In order to fully describe an activity, one needs to specify, besides its type (e.g., *Recreation*) the following attributes:

- *hasVenue*: This is a multivalue property (e.g., multiple shops to buy some good) defining the venue of the activity.
- *hasScheduledTime*: This property is used to represent the starting and ending times of an *already scheduled* activity. Its range is a *TimeInterval* object.
- *hasDuration*: This property defines the duration of the activity.
- *hasTemporalWindow*: This property defines the temporal domain of the activity, in the form of a list of temporal intervals (class *TimeInterval*). Scheduled activities need to have been scheduled within their domain.
- *transportationTypeToVenue*: This is a multivalue property defining the alternative means of transport that can be used in order to go to the venue.
- *transportationTypeFromVenue*: Similarly, this is a multivalue property defining the alternative means of transport that can be used to depart from the venue.
- *requires*: This is an important property defining what is a prerequisite in order to accomplish the activity. For example, it can define objects that the user should have or other actions that should have been already accomplished, in order to be able to accomplish the current activity, or even weather conditions. For example, in order to attend a movie film, the user should have bought a ticket; similarly, in order to run, the user should wear running shoes. This property has no defined range, since anything could be a prerequisite for an activity. Obviously, it is a multivalue property.

3.2 Punning

In order for OWL2 DL ([4], [13]) to remain decidable, it does not allow to define multiple types for any term (class, property, object, etc) of an ontology. In other words, it is not possible for an object, e.g., *OutdoorActivity*, to be defined both as a class, as well as the value of a property, that is an object (Individual). This is a significant constraint for the ontology, since it makes difficult to describe information concerning classes. For example, we want to describe the information that activity *Jogging_with_friends* will take place next Friday, which can be done by defining the particular object of the class *JoggingActivity*. Simultaneously, we also want to describe the information that one of the favorite type of activities for some user is *JoggingActivity*. In the first case the :*JoggingActivity* URI is used as a class name, whereas in the latter case it is used as a Named Individual.

To tackle this problem, we used the punning technique ([20]), which allows defining terms in an ontology, that have the same name but different types. This let us define both *JoggingActivity* class, as well as an object with the same name, in order to express that this is a favorite activity. Punning has been used in the two main classes (as well as all of their subclasses): *Activity* and *Venue*. 101 named Individuals share the same name (URI) with the corresponding classes of the ontology. This technique was introduced in our ontology to mainly avoid cluttering the namespace of the ontology with too many distinct URIs.

3.3 Examples

Let's try to describe a few examples, using the *Activities* ontology, in Turtle format:

Scenario 1: Nick is arranging a basketball match with two friends, to take place at the local public sports center on next Friday, from 17:00 to 19:00. He is going to the sports center with the bus plus some walking, but he will return with a taxi, so he needs some money to have with him. Of course, he needs a ball with him.

```
:Basketball_with_friends rdf:type :Sports ,
        owl:NamedIndividual,owl:Thing ;
   :hasStartTime "2012-09-
14T17:00:00"^^xsd:dateTime ;
   :hasEndTime "2012-09-14T19:00:00"^^xsd:dateTime
;
   :hasOtherParticipant :John , :Nick ;
   :transportationTypeToVenue :Bus ;
   :transportationTypeToVenue :OnFoot ;
   :transportationTypeFromVenue :Taxi ;
   :requires :Wallet ;
   :requires :Basketball_Ball ;
   :hasVenue :Sports_Center ;
:John   rdf:type :Person ,owl:NamedIndividual .
:Nick   rdf:type :Person ,owl:NamedIndividual .
:Wallet rdf:type owl:NamedIndividual ,owl:Thing ;
    rdfs:comment "My wallet to pay for the
taxi"@en .
:Basketball_Ball rdf:type     owl:NamedIndividual
,owl:Thing ;
    rdfs:comment "Basketball ball"@en .
:Poseidonio_mpasket rdf:type :BasketballCourt ,
        owl:NamedIndividual ;
   :hasName "Public Sports Center"^^xsd:string ;
   :hasLocation :posLocation .
:spLocation rdf:type :GeoLocation ,
        owl:NamedIndividual ;
    :hasLatitude "22.949476"^^xsd:double ;
    :hasLongitude "40.596664"^^xsd:double .
```

Scenario 2: Helen wants to go for shopping on Saturday. Near her home, there are two shopping centers "Mall A" and "Mall B" with different open hours. She will create a compound activity in her calendar, letting the digital assistant select the most appropriate mall to visit. A partial representation (due to limited space) of this activity can be defined as follows:

```
:VisitingMall rdf:type :ComplexDisjointActivity ,
        owl:NamedIndividual ;
    :hasPart :Visiting_mall_A ,
    :Visiting_mall_B .
:Visiting_mall_A rdf:type :HumanActivity ,
        owl:NamedIndividual ;
    :hasTemporalWindow :mall_A_open_hours .
```

```
:Visiting_mall_B rdf:type :HumanActivity ,
        owl:NamedIndividual ;
    :hasTemporalWindow :mall_B_open_hours .
:mall_A_open_hours rdf:type :TimeInterval ,
        owl:NamedIndividual ;
    :hasEndTime :instant_A_end ;
    :hasStartTime :instant_A_start .

:mall_B_open_hours rdf:type :TimeInterval ,
        owl:NamedIndividual ;
   :hasEndTime :instant_B_end ;
   :hasStartTime :instant_B_start .
:instant_A_end rdf:type owl:NamedIndividual ,
        time:Instant ;
   time:inXSDDateTime "2014-12-
06T09:00:00"^^xsd:dateTime .
:instant_A_start rdf:type owl:NamedIndividual ,
        time:Instant ;
   time:inXSDDateTime "2014-12-
06T15:00:00"^^xsd:dateTime .
:instant_B_end rdf:type owl:NamedIndividual ,
        time:Instant ;
   time:inXSDDateTime "2014-12-
06T12:00:00"^^xsd:dateTime .
:instant_B_start rdf:type owl:NamedIndividual ,
        time:Instant ;
   time:inXSDDateTime "2014-12-
06T20:00:00"^^xsd:dateTime .
```

4. THE *MEETINGS* ONTOLOGY

The *Meetings* Ontology aims to represent semantically all the information needed in order for two or more people to arrange a meeting, thus reducing the need for complex interactions towards reaching an agreement. So, the most important is to represent the available time of every person, as well as the meeting request details (e.g., range, duration, alternative locations, potential participants). The *Meetings* ontology comprises:

- 22 Classes (owl:Class)
- 26 Named Objects (owl:NamedIndividual)
- 25 Object Properties (owl:ObjectProperty)
- 2 Data Properties (owl:DataProperty)
- 27 Subclass Axioms (rdfs:subClassOf)
- 13 Equivalent Classes (owl:equivalentClass)

4.1 The *Meeting* class

The basic class of the ontology is *Meeting*, with only two subclasses, *ArrangedMeeting* and *ToBeArrangedMeeting*, with the obvious meaning.

```
##http://www.uom.gr/ai/ontologies/Meetings.owl#Mee
ting
:Meeting rdf:type owl:Class ;
        rdfs:subClassOf
        [
                rdf:type owl:Restriction ;
                owl:onProperty :topic ;
                owl:someValuesFrom xsd:string
        ] , [
                rdf:type owl:Restriction ;
                owl:onProperty :hasParticipant ;
                owl:someValuesFrom :Participant
        ] , [
                rdf:type owl:Restriction ;
                owl:onProperty :hasLocation ;
                owl:someValuesFrom            owl-
                time:Location
        ] , [
                rdf:type owl:Restriction ;
                owl:onProperty :name ;
                owl:someValuesFrom xsd:string
        ] ;
        owl:disjointUnionOf (
                :ArrangedMeeting
                :ToBeArrangedMeeting
        ) .
```

Each meeting may have one or more topics (property *topic*), as well as one or more locations (property *hasLocation*). Object *NoLocation* denotes that the meeting is not bound to a specific location, e.g., in case of making a mobile phone call. Property *hasParticipant* denotes the tentative participants of the meeting.

Arranged meetings have the properties *hasStartTime*, *hasEndTime* and *hasDuration*. On the other hand, not yet arranged meetings have the multivalue property *hasTemporalWindow*, allowing to define temporal intervals (windows) when the meeting can be scheduled. They also have a minimum (property *hasMinDuration*), a maximum (property *hasMaxDuration*) and a preferred (property *hasPreferredDuration*) duration.

4.2 The *Participant* Class

The *Participant* class aims at representing the participants of a meeting. For each participant, we need to know its role in the meeting (multivalue property *hasRole*), his importance (*hasPresenceType*) and his availability (property *hasTimeProgram*).

There are five roles for the participants, defined as objects of the class *ParticipantRole*. These are *Chairman*, *Organizer*, *Member*, *Guest* and *NoRole*. Similarly, there are five types of presence for any participant, namely *RequiredPresence*, *DesiredPresence*, *PermittedPresence*, *NoPresence* and *ProhibitedPresence*.

4.3 The *TimeProgramOrganizer* class

The *TimeProgramOrganizer* class is used to describe all the necessary information concerning availability and non-availability of any participant. Three multivalue properties are defined over this class: *hasCalendar*, ranging over *Calender* objects, which denotes the available user's calendars and the events they already contain; *hasExcludedInterval* and *hasPreferredInterval* ranging over *Interval* or *ComboInterval* objects, with the former denoting the temporal intervals when the user is unavailable (although, according to his calendars, he has nothing else scheduled at this time) and the latter denoting the available intervals that are more desirable by the user.

4.4 The *Calendar* and *CalendarEntry* classes

A *Calendar* object represents available and non-available time windows over a specific time period; it does not contain information about specific activities.

```
##http://www.uom.gr/ai/ontologies/Meetings.owl#Cal
endar
:Calendar rdf:type owl:Class ;
        owl:equivalentClass [
                rdf:type owl:Restriction ;
                owl:onProperty :hasEntry ;
                owl:someValuesFrom :CalendarEntry
        ] .
```

A time window is represented with a *CalendarEntry* object, with the following properties defined over it: *hasDuration* (number), *hasStartTime* (date), *hasEndTime* (date), *hasImportanceType* (ranging over *ImportanceType*), *hasMovabilityType* (ranging over *MovabilityType*) and *hasType* (ranging over *EntryType*).

There are five levels of importance, defined as objects of the class *ImportanceType*, namely *Trivial*, *SlightlyImportant*, *Important*, *FairlyImportant* and *Crucial*. Less important activities may get eliminated from the calendar by an ICA scheduling engine, in favor of more important activities, in case not enough time is available.

Similarly, there are five levels of movability, having to do with whether an activity can be moved from its current scheduled position to another time window: *TotallyImmovable*, *SlightlyImmovable*, *Movable*, *FairlyMovable* and *FullyMovable*. For example, an activity like *AI_class* may be characterized as *SlightlyImmovable*, whereas an activity like *Biking* may be characterized as *FairlyMovable*. Again, this semantic information is to be considered by the scheduling engine of an ICA.

A calendar entry may be one of the following two types: *Available* or *Busy*. Calendar entries of the type *Available* have the meaning that the user has something to do at that time, but he is also available to do other things as well. On the other side, calendar entries of the form *Busy*, have the meaning that nothing else can be scheduled in parallel to the busy calendar entry.

4.5 The *Interval* and *ComboInterval* classes

These two classes are used as the range of the *hasPreferredInterval* and *hasExcludedInterval* properties, which are defined over *TimeProgramOrganized* objects. They are used to represent temporal intervals. *ComboInterval* represents lists of intervals, whereas *Interval* represents simple intervals.

```
##http://www.uom.gr/ai/ontologies/Meetings.owl#Interval

:Interval rdf:type owl:Class ;
        owl:equivalentClass [
                rdf:type owl:Class ;
        owl:intersectionOf (
        [
                rdf:type owl:Restriction ;
                owl:onProperty :endsAt ;
                owl:someValuesFrom owl:Thing
        ]
        [       rdf:type owl:Restriction ;
                owl:onProperty :startsAt ;
                owl:someValuesFrom owl:Thing])
        ] ;
        owl:disjointUnionOf (
                :DateInterval
                :DayInterval
                :TimeInterval ) .

###
http://www.uom.gr/ai/ontologies/Meetings.owl#ComboInterval

:ComboInterval rdf:type owl:Class ;
        owl:equivalentClass [
                rdf:type owl:Restriction ;
                owl:onProperty :consistsOf ;
                owl:someValuesFrom :Interval
        ] .
```

Subclass *DateInterval* represents intervals with absolute dates, e.g., December 20th, 2012 to December 25th, 2012; subclass *DayInterval* represents intervals between dates, e.g., Monday to Friday, without an absolute reference to specific dates; and, finally, subclass *TimeInterval* represents intervals between hours within a day, e.g., from 10:00 to 18:00, without a specific day or date. The type of the properties *hasStart* and *hasEnd* changes according to the specific subclass.

4.6 Examples

With the above ontologies, we can represent scenarios like the following:

Scenario 1: A user prefers his meetings to be scheduled, whenever possible, on Mondays or Tuesdays, before the lunch break (which is at 15:30). On Saturdays and Sundays he does not want to schedule any meeting. Furthermore, he does not want to schedule a meeting during the Christmas period (23/12/2014 to 03/01/2015).

Scenario 2: A user's calendar, on November 20th, contains two activities: (a) Lecture at the AI class, from 12:00 to 15:00, and (b) biking, from 17:00 to 18:00.

Scenario 3: The meeting has arranged for January 7th, 2015, from 12:00 to 14:30, in Thessaloniki, between Kostas, Bill and Helen.

Scenario 4: A meeting has to be arranged between John, Maria and Nick, in the week between January 12th and January 16th, 2015. The presence of John and Maria is required, whereas the presence of Nick is just desired. The meeting will take place through Skype, so anyone can be at his own place. The duration of the meeting is at least 2 hours and at most 3,5 hours.

Due to the limited space, we provide the representation of these four scenarios using the Meetings Ontology in the following link: https://github.com/agnantis/uom-ontologies/tree/master/Meetings/examples.

5. THE *CALENDARPREFS* ONTOLOGY

The *CalendarPrefs* ontology aims at representing details about the user of an ICA, as well as his preferences over the way activities should be scheduled within his calendar. The ontology comprises:

- 37 classes (owl:Class)
- 12 named Objects (owl:NamedIndividual)
- 19 object properties (owl:ObjectProperty)
- 6 data properties (owl:DataProperty)
- 55 subclass axioms (rdfs:subClassOf)
- 20 equivalent classes (owl:equivalentClass)

The basic classes of the ontology are two, namely *ActivityPreference* and *UserProfile*, with several other auxiliary classes. We will proceed with the description in a bottom-up direction, starting from the auxiliary classes.

- Class *Topic* is used to represent the interests of a user, such as *Technology*, *Health* or *Athletics*.
- Class *EducationLevel* is an enumerated class that represents the educational level of the user. Its values (i.e.

named individuals) are *Primary*, *Secondary*, *College*, *University*, *MScHolder* and *PhDHolder*.

- Class *ActivityExecutionType* is an enumerated class consisting of two members: *NotPreemptiveExecution*: indicating activities that can only be executed at once, and *PreemptiveExecution*, representing activities that can be executed in parts.
- Class *ActivityType* represents activity types. It can be seen as a connection point between this ontology and *Activities* ontology, since an individual of this class can also be an individual of the *Activity* class.
- Enumerated class *Gender* has two values, *Male* and *Female*.
- Class *LocationType* represents either specidic locations (e.g., a user's home) or location types (e.g., football court). A member of the *LocationType* class can also be a member of *Venue* class.
- Class *Preference* is a general class that represents all sorts of preference. *ActivityPreference* is a subclass of it.
- Class *Profile* is a general class that represents all sorts of profiles. Class *UserProfile* is a subclass of it.

5.1 The *UserProfile* class

This class represents the personal details of the user. This is achieved through the following properties, ranging over simple data types (strings and numbers) or over the auxiliary aforementioned classes, and having the obvious interpretation:

hasFirstName, *hasLastName*, *hasNickName*, *hasGender*, *hasAge*, *hasEmail*, *hasEducationBackground*, *hasInterestIn*, *hasCurrentWork*, *hasPreviousWork*, *homeLocation*, *workLocation* and *hasActivityPreference*.

5.2 The *ActivityPreference* class

The *ActivityPreference* class has several subclasses, depending on the type of preference over activities. The two subclasses, *UnaryPreference* and *BinaryPreference*, define preferences that concern single activities and pairs of activities respectively. The three subclasses *ModePreference*, *LocationPreference* and *TemporalPreference* define preferences that concern the mode in which the activity is executed, the location it is scheduled and the temporal way it is scheduled respectively.

A subclass of *ModePreference* is *ActionPreference*, defining whether an activity is executed in preemptive or non-preemptive mode.

LocationPreference has two subclasses, *PreferLocation* and *AvoidLocation*, defining types of locations that the user prefers or does not prefer an activity to be scheduled there. For example, a user prefers to play basketball in closed halls, whereas he does not prefer shopping in malls.

TemporalPreference has several subclasses, defining how an activity is scheduled in time. Particularly:

- *MaxFrequency*: This class is used to define the maximum repetitions of an activity over a temporal interval. For example, the user might want to go at most 3 times a week to the gym.
- *MinFrequency*: This class defines the minimum number of repetitions of an activity over a temporal interval. For example, the user might want to go at least 2 times a month to the gym.
- *PreferredFrequency*: This class defines the preferred number of repetitions of an activity over a temporal interval.
- *MaxTimeDistance* / *MinTimeDistance* / *PreferredTimeDistance*: These three classes define the maximum / minimum / preferred temporal distance between the end time of an activity and the start time of another.
- *MaxAccumulatedDuration* / *MinAccumulatedDuration* / *PreferredAccumulatedDuration*: These classes define the maximum / minimum / accumulated duration of an activity.
- *MaxPartDuration* / *MinPartDuration* / *PreferredPartDuration*: The maximum / minimum / preferred duration of each part of a preemptive activity. This preference has to do only with preemptive activities, otherwise it is ignored.

5.3 Examples

Using the *CalendarPrefs* ontology, we can represent the knowledge of the following scenarios:

Scenario 1: If a social activity is followed by a sports activity, we prefer to have 3 hours temporal distance between them but in no case less than 2 hours.

```
###
http://www.uom.gr/ai/ontologies/CalendarPrefs.owl#
timeDistPreference
:timeDistPreference                          rdf:type
:TimeDistancePreference ,
owl:NamedIndividual ;
:hasFirstMember activities:SocialActivity ;
:hasSecondMember activities:SportsActivity ;
:hasMinDuration :durationA ;
:hasPrefDuration :durationB .

###
http://www.uom.gr/ai/ontologies/CalendarPrefs.owl#
durationA
:durationA rdf:type owl:NamedIndividual ,
owl-time:DurationDescription ;
owl-time:hours ”2”^^xsd:nonNegativeInteger .

##http://www.uom.gr/ai/ontologies/CalendarPrefs.ow
l#durationB
:durationB rdf:type owl:NamedIndividual ,
       owl-time:DurationDescription ;
       owl-time:hours  ”3”^^xsd:nonNegativeInteger
.
```

Note that although object *timeDistancePreference* belongs to class *TimeDistancePreference*, a OWL-reasoner is able to infer that it also belongs to *MinTimeDistance* and *PrefTimeDistance*.

Here is another scenario that can be represented by the same ontology.

Scenario 2: Just before each football match, it is necessary to have 30 minutes light running, and just after the football match it is necessary to have 15 minutes stretching exercises.

```
##http://www.uom.gr/ai/ontologies/CalendarPrefs.ow
l#impliesPrefA
:impliesPrefA rdf:type :TimeDistancePreference ,
              owl:NamedIndividual ;
       :hasFirstMember
activities:FootballActivity ;
       :hasSecondMember :shortJoggingSession .
       :hasOrder :follows ;
       :hasMaxDuration :5MinDuration ;
##http://www.uom.gr/ai/ontologies/CalendarPrefs.ow
l#impliesPrefB
:impliesPrefB rdf:type :TimeDistancePreference ,
              owl:NamedIndividual ;
       :hasFirstMember
activities.owl#FootballActivity ;
       :hasSecondMember :shortStretchingSession .
       :hasOrder :pr  oceeds ;
       :hasMaxDuration :5MinDuration ;

##http://www.uom.gr/ai/ontologies/CalendarPrefs.ow
l#5MinDuration
:5MinDuration rdf:type owl:NamedIndividual ,
              owl-time:DurationDescription ;
       owl-time:minutes
"5"^^xsd:nonNegativeInteger .

##http://www.uom.gr/ai/ontologies/CalendarPrefs.ow
l#shortJoggingSession
:shortJoggingSession rdf:type
              activities:JoggingActivity ,
       owl:NamedIndividual ;
       activities:hasDuration :joggingDuration .
##http://www.uom.gr/ai/ontologies/CalendarPrefs.ow
l#joggingDuration
:joggingDuration rdf:type owl:NamedIndividual ,
              owl-time:DurationDescription ;
       owl-time:minutes
"30"^^xsd:nonNegativeInteger .

##http://www.uom.gr/ai/ontologies/CalendarPrefs.ow
l#shortStretchingSession
:shortStretchingSession rdf:type
              activities:GymnasticsActivity ,
              owl:NamedIndividual ;
       activities:hasDuration  :stretchingDuration
.

##http://www.uom.gr/ai/ontologies/CalendarPrefs.ow
l#stretchingDuration
:joggingDuration rdf:type owl:NamedIndividual ,
              owl-time:DurationDescription ;
       owl-time:minutes
"15"^^xsd:nonNegativeInteger .
```

6. RELATED WORK

Raimond and Abdallah [19] describe an Event Model, where the event is defined by its location, time, participants/agents, factors and products. Factor and product are general properties that are used to represent any aspect or output of an event respectively, and as their range is owl:Thing, their semantic impact is quite limited.

Similarly, the Simple Event Model proposed by Willen and Davide [14] tries to provide a simple RDF model for describing any event based on its type (what), its location (where), its participants (who) and its temporal information (when), allowing for a large category of questions to be answered using simple SPARQL queries. Our Activity ontology, tries to describe similar aspects of an event, focusing also on the definition of more specific properties and event types, aiming to achieve better semantic interpretation of the related information for Intelligent Calendar Applications.

Riboni D. and Bettin C. [18] present an Activity ontology similar to our work. Having the Activity class as the root of the ontology, they define a hierarchy of different sub-activities based on specific rules that an activity applies to, e.g., the number of participants (SocialActivity), or various characteristics of them (e.g CarnivalParty). As their total work is more interested in identifying indoor complex human activities, their ontology is mainly focused on modeling activities for smart home and sensor information.

Another similar approach was followed in the EU Project Dem@Care ([11]), where the Activity ontology tries to represent simple every day human activities or conditions (like Sleeping or In the Bathroom) and by composing them to produce new more interesting and complex ones.

Finally, Yingjie H. and Janowicz K. [9] propose a quite interesting and promising idea of defining various Activity Ontologies, which will be able to interconnect human activities in the real world (like writing a paper) with computer events in the digital world (e.g. editing software or calendar events).

7. INTEROPERABILITY WITH OTHER ONTOLOGIES

Two of the main principles of ontology development are reusability and interconnection between ontologies. They both constitute key features for the broad acceptance of an ontology by users, systems and applications. For that reason, we developed and keep updating our ontologies to use existing ontologies when it is possible. Currently our ontologies import and use elements of the following well known ontologies:

- *FOAF ontology* [5], which is used to represent user related information, like names, addresses, contact info etc.
- *OWL-Time* ontology [7], which is used to represent time related information, like duration, dates, intervals etc.
- *Geo OWL ontology* [23], a simple ontology/vocabulary proposed by W3C to describe spatial information, such as the latitude or the longitude of a specific location on earth.

Even though we import the *Geo* OWL ontology, we decided to define our own *GeoLocation* class (in *Activities* ontology) in order to represent geo-spatial information. The intuition behind this design choice to introduce a new class (i.e., *GeoLocation*) instead of using an existing one provided by other geospacial ontologies, like *Geo OWL* [23], *WGS84 Geo Positioning* [22], or *GeoRSS* [21], lies on the fact that there seems to be no consensus on a generally approved ontology for representing geospatial information [24].

However, the two data properties *hasLatitude* and *hasLongitude* of the *GeoLocation* class, are connected - as equivalent properties - with the *latitude* and *longitude* properties of the *Geo OWL* ontology, in order to create a relationship between the two ontologies, mainly for future reference, as the *Geo OWL* ontology seems the most prominent and feature-rich one among other geospatial ontologies.

8. CONCLUSIONS AND FUTURE WORK

In this paper we presented three ontologies, all having to do with the representation of arbitrary types of everyday activities, including meeting arrangements and a wide range of preference modeling. The goal besides these ontologies is to be used by the emerging intelligent calendar applications.

Particularly, the *Activities* ontology aims at describing various types of activities, giving them semantical context. So, an Intelligent Calendar Application could present to the user a hierarchy of activities, thus allowing him easily to identify the one he wants to insert to his calendar. Additionally, other sites, e.g., a shopping center, could provide ready-to-insert descriptions of the activity to visit them, including open hours, location, alternative ways to reach them, etc.

The *Meetings* ontology helps the users of an intelligent calendar application to arrange meetings in an automated or semi-automated way, taking into account their availability, their preferences over the various time windows, their role and their importance for the meeting, etc. Arranging a meeting may involve rearranging other already scheduled activities.

Finally, the *CalendarPrefs* ontology aims at expressing a user's preferences over the way he prefers to schedule activities of specific classes. This ontology is to be used internally by an ICA, thus the user has not to re-enter his references over specific activities each time he requests a new plan.

For the future, besides extending these ontologies with new features, we are working on incorporating them within a real Intelligent Calendar Application. All three ontologies and the examples presented in this paper can be found in the following public GitHub repository: https://github.com/agnantis/uom-ontologies.

9. ACKNOWLEDGMENTS

This research has been co-financed by the European Union (European Social Fund – ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Heracleitus II. Investing in knowledge society through the European Social Fund.

10. REFERENCES

[1] Alexiadis, A. & Refanidis, I.: Optimizing Individual Activity Personal Plans through Local Search. Accepted for publication by *AI Communications*, an ECCAI journal (2015)

[2] Alexiadis, A. & Refanidis, I.: Post-Optimizing Individual Activity Plans through Local Search. In Proceedings of ICAPS 2013 Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems (COPLAS), pp. 7-15. Rome (2013)

[3] Alexiadis A. & Refanidis I.: Defining a task's temporal domain for intelligent calendar applications. In AIAI, pp. 399-406 (2009)

[4] Baader F., Horrocks I., Sattler U.: Description Logics, in: The Handbook on Ontologies in Information Systems, 2nd edition, S. Staab, R. Studer (eds.), Springer Verlag, (2009)

[5] Brickley, D. and Miller, L.: "FOAF vocabulary specification 0.98." Namespace document 9 (2012).

[6] Gkekas G., Kyrikou A., Ioannidis N.: A smart calendar application for mobile environments. In Proceedings of the 3rd international conference on Mobile multimedia communications, ICST, Brussels, Belgium (2007)

[7] Hobbs, Jerry R., and Feng Pan. "An ontology of time for the semantic web." *ACM Transactions on Asian Language Information Processing (TALIP)* 3.1 (2004): 66-85.

[8] Horrocks, I. "What Are Ontologies Good For?" In Evolution of Semantic Systems, pp. 175-188. Springer, Heidelberg (2013)

[9] Hu, Y., & Janowicz, K. (2012, November). Improving personal information management by integrating activities in the physical world with the semantic desktop. In Proceedings of the 20th International Conference on Advances in Geographic Information Systems (pp. 578-581). ACM.

[10] Refanidis I. & Alexiadis A.: SELFPLANNER: Planning your Time! ICAPS 2008 Workshop on Scheduling and Planning Applications, Sydney (2008)

[11] Meditskos, G., Dasiopoulou, S., Efstathiou, V., & Kompatsiaris, I. (2013). Ontology patterns for complex activity modelling. In Theory, Practice, and Applications of Rules on the Web (pp. 144-157). Springer Berlin Heidelberg.

[12] Mizoguchi R., Bourdeau J.: Using Ontological Engineering to Overcome Common AI-ED Problems. Journal of Artificial Intelligence and Education, 11, pp.107-121 (2000)

[13] Owl 2 web ontology language document overview. http://www.w3.org/TR/owl2-overview/

[14] Raimond, Y., & Abdallah, S. (2007). The event ontology. Technical report, 2007. http://motools. sourceforge. net/event.

[15] Refanidis I., Alexiadis, A.: Deployment and Evaluation of SelfPlanner, an Automated Individual Task Management System. *Computational Intelligence*, 27(1) 41-59 (2011)

[16] Refanidis I., Alexiadis A., Yorke-Smith N.: Beyond calendar mashups: SELFPLANNER 2.0, pp. 66-70 (2011)

[17] Refanidis, I., Yorke-Smith, N.: A Constraint Based Programming Approach to Scheduling an Individual's Activities. *ACM Transactions on Intelligent Systems and Technologies*, vol. 1 (2) (2010)

[18] Riboni, D., & Bettini, C. (2011). OWL 2 modeling and reasoning with complex human activities. Pervasive and Mobile Computing, 7(3), 379-395.

[19] Van Hage, W. R., Malaisé, V., Segers, R., Hollink, L., & Schreiber, G. (2011). Design and use of the Simple Event Model (SEM). Web Semantics: Science, Services and Agents on the World Wide Web, 9(2), 128-136.

[20] Vrandecic D., Krotzsch M.: Reusing Ontological Background Knowledge in Semantic Wikis, Proceedings 1st Workshop on Semantic Wikis. Budva, Montenegro, June (2006)

[21] http://www.georss.org/

[22] http://www.w3.org/2003/01/geo/wgs84_pos

[23] http://www.w3.org/2005/Incubator/geo/XGR-geo/#owl

[24] http://www.w3.org/2005/Incubator/geo/XGR-geo-ont-20071023/#ontologies