Semantic Explorable Representation of 3D Content Behavior

Jakub Flotyński(✉), Mikołaj Maik, Paweł Sobociński, and Michał Śliwicki

Poznań University of Economics and Business, Poznań, Poland
{flotynski,maik,sobocinski,sliwicki}@kti.ue.poznan.pl
http://www.kti.ue.poznan.pl/

Abstract. Extended reality (XR) environments are successfully used in various domains, such as medicine, education, training, and industry. The essential element of every XR environment, apart from the XR/3D content, is the domain knowledge encoded in the environment, which is expressed by the behavior of the content, including its autonomous actions and interactions with the XR users. Such knowledge covers the processes occurring in the environment, characteristics of its objects, and skills of the users. However, the available approaches to XR creation and usage focus on the sensory perception of XR, including its visual, haptic, and aural elements, and knowledge acquisition using human senses, but they do not enable knowledge acquisition based on automated reasoning and queries. It strongly limits the possibilities of knowledge discovery from XR, in particular on a massive scale in web based environments and repositories. In this paper, we propose a knowledge-based model to 3D content behavior representation. The model permits generic representation of temporal properties of XR environments with domain terminology, which can be subject to semantic exploration with queries and reasoning. Our solution may contribute to broader and more efficient dissemination of knowledge using XR.

Keywords: Extended reality · Knowledge · Exploration · Temporal

1 Introduction

Extended reality (XR) is increasingly used in many companies and institutions. They introduce XR systems to improve their basic operations, such as employee training, merchandising, product design, marketing, preserving cultural heritage, and education. XR environments are built using multiple 3D objects. While using an XR environment, its 3D objects change their properties—states, appearance, and geometry—in response to user interactions, mutual interactions with other objects as well as autonomous actions. Such interactions and actions constitute the XR behavior and express the knowledge in the domain for which the environment has been developed. However, currently, the acquisition of knowledge by users is possible mostly by immersing into the environment and directly interacting with its 3D content.

Actions and interactions within XR environments could also be subject to exploration based on semantic queries and automated reasoning. Such exploration has the potential to enrich various application domains, especially in the context of the collaborative network- and web-based XR providing additional information about users' behavior, e.g., past, current, and possible future actions.

Furthermore, such exploration can be used as a method for analyzing and improving the user experience. By using the exploration of content behavior, we can increase the understanding of the relationship between objects in the scene, the user and their interactions. Exploration can provide an efficient tool for measuring the user experience, for example, which interactions in the scene were the most difficult and time-consuming, and the differences in interaction with different objects. Such information can be used for evaluating the system to improve the user experience. Additionally, semantic queries can be used to predict the activity in the virtual scene, so the system can be adjusted for the user to improve the quality of use.

For the possibility of this kind of exploration, XR environments need to use an appropriate representation for interactive 3D content behavior. That can be created using knowledge representation technologies such as the semantic web standards (the Resource Description Framework—RDF [19], the RDF Schema—RDFS [20], the Web Ontology Language—OWL [18] and the SPARQL query language [17]) and ontologies, which gain increasing attention in 3D modelling and multimedia description [14,15].

The main contribution of this paper is a knowledge-based model for 3D content behavior representation. The primary goal of the model is to enable the expression of behavior semantics in any domain, regardless of the specific XR environment's 3D graphics and animation technologies. The proposed model consists of two sub-models responsible for the representation of different types of behavior using different knowledge representation technologies: the ontology-based component and the rule-based component. The model uses Semantic Web standards - RDF, RDFS and OWL.

The proposed model has been used to represent the behavior of the virtual environment in the industrial XR training system developed for Amica S.A (Poland's main house appliances manufacturer) to train employees how to operate specialized industrial devices.

The remainder of the paper is structured as follows: Sect. 2 provides an overview of the current state of modelling 3D content behavior. Then, Sect. 3 explains the proposed semantic model of 3D content behavior. Section 4 provides an example and overview of the generated knowledge base for the XR training system. Finally, Sect. 5 concludes the paper and indicates possible future research.

2 Related Work

The area of semantic representation of 3D content behavior has gained little attention in the scientific literature and is not present in the available XR environments. So far, ontologies have been used mostly for the representation of the structure, geometry, appearance, and animation of 3D content, which has been summarized in [6].

A few approaches have been developed to model the behavior of 3D content using ontologies and semantic web standards. One such approach, proposed in [9,10,12], provides temporal operators and allows for the expression of both primitive and complex behavior. Based on this approach, a graphical tool has been implemented to model complex behavior using diagrams, with the ability to encode the specified behavior in X3D scenes [11]. In another approach, presented in [5], primitive actions such as move, turn and rotate are combined to represent complex behavior in a manner that is easily understandable to end users, without requiring knowledge of 3D graphics and animation

In [8], a tool that utilizes semantic concepts, services, and hybrid automata to describe the behavior of 3D content elements is presented. The client is based on a 3D content presentation tool, such as an XML3D browser, while the server is composed of various services that facilitate content selection and configuration of 3D content. Additionally, a separate module is responsible for managing intelligent avatars, including their perception of the scene.

In [1], XSD-based semantic metadata schemes were proposed for describing the interactivity of 3D objects using events, conditions, and actions. Ontologies presented in [3,4] provide a means of representing multi-user virtual environments and avatars. These ontologies define the geometry, space, animation, and behavior of 3D content, with concepts that are semantic equivalents to those used in commonly-used 3D content formats such as VRML and X3D. Environmental objects, the main entities of 3D content, are described using translation, rotation, and scale. Avatars, on the other hand, are described using names, statuses, and user interfaces (UIs), and their behavior is described using code bases.

In the academic literature, researchers have proposed several approaches for spatiotemporal reasoning on semantic descriptions of evolving human embryos and 3D molecular models. For instance, in [13], the authors proposed an approach that leverages RDF, OWL, and SPARQL, along with an ontology that describes stages, periods, and processes. In a similar vein, [16] proposed an approach that combines different input and output modalities to enable presentation and interaction tailored to specific types of content and tasks. While there has been more research on the semantic representation of animations than interactions, the existing approaches in both areas remain preliminary and face several challenges, particularly in terms of content exploration

There is also another recent work that focuses on humans and their interactions in virtual reality and on creating an ontology for experimentation of human-building interaction that uses virtual reality (VHBIEO) [2]. The proposed solution aims to create a standardized approach to virtual human-building interaction experimentation. To achieve this, an ontology has been developed that builds on existing ontologies and semantic models, such as EXPO, STED, DNAs, ifcOWL, SSN, SUR, and UO. The DOGMA methodology has been employed to establish the internal structure of the ontology. In a similar vein, another ontology has been introduced in [7]. This ontology focuses on human-centred analysis and design of virtual reality conferencing, with goals that include enhancing user experience, facilitating research on VR-conferencing (especially psychological and behavioral), and enabling the sharing of research findings. Although both ontologies are still in their early stages of development.

3 Semantic Behavior Model

The presented knowledge-based model is a formal representation of 3D objects' behavior, including their interactions (with users and other objects) and autonomous actions. Knowledge exploration is made possible through the use of complex domain terminology, which is tailored to a specific application domain. The model may be used for any domain ontology.

The model allows for the representation of the behavior of XR objects and environments. The behavior of an XR environment encompasses the actions of its users and objects. The model consists of two main components based on distinct knowledge representation technologies, which differ in expressiveness.

1. The ontology-based behavior representation component provides a foundation for representing the behaviors of explorable XR environments. This component encompasses classes and properties defined in ontologies based on Semantic Web standards that implement description logic. The component consists of three main elements:
   (a) Domain ontologies, which are formal specifications of the conceptualization of individual application domains for which explorable XR environments are created.
   (b) A fluent ontology, which defines domain-independent classes and properties of temporal entities and the relationships between them. The ontology also specifies entities for the visual representation of the behaviors of XR components and environments. Additionally, it enables visualization associated with knowledge exploration.
   (c) Visual semantic logs, which represent knowledge about the behavior of users and objects demonstrated during sessions of using explorable XR environments.
2. The rule-based component for behavior representation provides complex relationships between temporal entities defined in the fluent ontology component. In particular, this component enables transitions between events and states that are essential for describing the behavior of XR components and environments. Entities and relationships are specified in a fluent set of rules. The rule set is defined using logic programming, which allows for describing relationships between the properties of individual objects.

The model utilizes knowledge representation technologies to provide axioms that allow for the representation of behaviors, features of components, and XR environments. An XR component is a reusable module, which is a collection of classes with methods that can be combined with other XR components in an XR environment.

3.1 Ontology-Based Component for Behavior Representation

This component is part of the behavior model used for representing the features as well as past and current behaviors of XR environments in the form of behavior logs, which encompass the actions of users and objects described using temporal statements. This component is based on Semantic Web standards - RDF, RDFS, and OWL.

States and Events. The fundamental units representing the behavior of explorable XR environments are states. A state is a representation of a fragment of the XR environment, specifically a set of concepts describing users and objects. States are described using terms, while their occurrence can be evaluated using predicates. In the presented method, we assume that an *Instantaneous State* is a state that occurs at a specific point in time, while an *Interval State* is a state that occurs over a time interval.

States are initiated and terminated by events. Like states, events are denoted using terms, while their occurrence can be evaluated using predicates.

1. The *begins* predicate is used to indicate that an event initiates a state. This predicate is true for a given event and state if and only if the event starts the state. It is denoted as *begins(event, state)*, for example, *begins(startOfRun, run)*.
2. The *finishes* predicate is used to indicate that an event terminates a state. This predicate is true for a given event and state if and only if the event ends the state. It is denoted as *finishes(event, state)*, for example, *finishes(endOfRun, run)*.

Fluent Ontology. The fluent ontology defines fundamental temporal entities - classes and properties - that allow for the representation of users' and objects' behaviors. The fluent ontology extends domain ontologies with temporal terminology, which can be used in conjunction with domain-specific classes and properties. The fluent ontology is an invariant part of the approach, which is common to all explorable XR environments, regardless of individual application domains.

1. The predicate *start* is used to indicate that a temporal entity begins at a specific point in time. This predicate is true for a given temporal entity and time point if and only if the temporal entity starts at that time point. It is denoted as *start(temporalEntity, tp)*.
2. The predicate *end* is used to indicate that a temporal entity terminates at a specific point in time. This predicate is true for a given temporal entity and time point if and only if the temporal entity ends at that time point. It is denoted as *end(temporalEntity, tp)*.

The fluent ontology defines a variety of distinct entities, such as:

1. *State:* An OWL class encompassing all states.

2. *InstantState:* A subclass of *State* representing all instantaneous states.
3. *IntervalState:* A subclass of *State* representing all interval states, which is disjoint from *InstantState*.
4. *Event:* An OWL class that includes all events.
5. *TemporalEntity:* An RDFS class that covers all temporal entities.
6. *TimePoint:* A subclass of both RDFS *datatype* and *TemporalEntity*, representing all time points. The use of RDFS *datatype* allows for specifying time domains according to the requirements of individual applications.
7. *TimeInterval:* A subclass of *TemporalEntity*, among others, that constitutes all time intervals. TimeInterval is described by two properties, *start* and *end*, which indicate the time points at which the interval begins and ends, respectively.
8. *TimeSlice:* An OWL class representing all time slices, each of which possesses two obligatory properties determined by qualified cardinality restrictions in OWL.
9. *VisualDescriptor:* An OWL class encompassing all visual descriptors.

Domain ontologies and fluent ontologies are used to create visual semantic behavior logs consisting of temporal statements. The concept of temporal statements is illustrated in Fig. 1. In the figure, every pair of nodes connected by a predicate represent a single RDF statement.

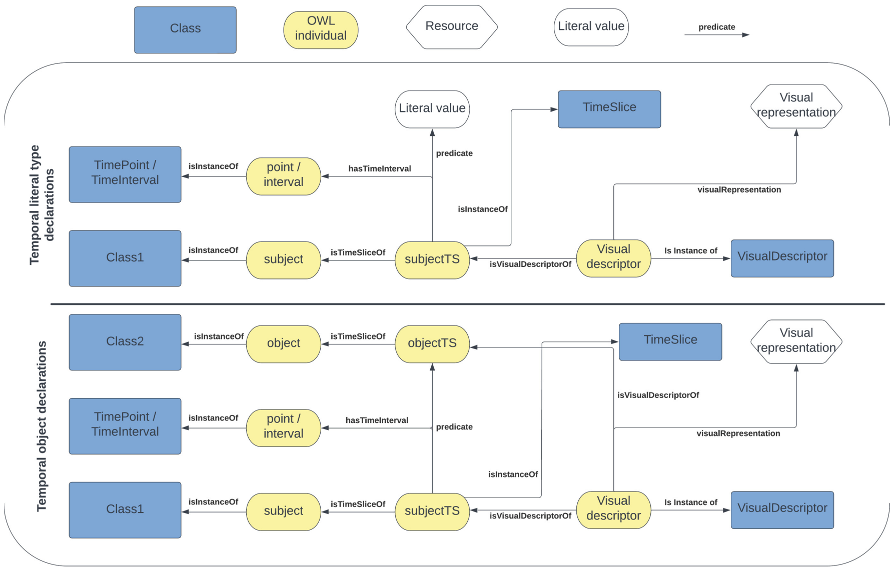


Fig. 1. Time statements (yellow) of the visual semantic behavior log. Entities that are related but not part of the time statements are highlighted in blue. (Color figure online)

Temporal Reasoning. We also define predicates for time points and intervals based on event calculus predicates and time ontology. Such predicates allow for temporal reasoning on time points and intervals.

– The predicate *in* is a predicate that is true for a given time point and time interval if and only if the time point is greater than or equal to the time point that starts the time interval and less than or equal to the time point that ends the time interval.

As examples of relationships between time intervals, we define three predicates from the Time Ontology proposed by Allen and Ferguson [18]: *before*, *after*, and *starts*.

– The predicate *before* is a predicate that is true for a given time interval ti1 and time interval ti2 if and only if ti1 ends before ti2 begins. *before(ti1, ti2)* is equivalent to *after(ti2, ti1)*.
– The predicate *starts* is a predicate that is true for a given time interval ti1 and time interval ti2 if and only if ti1 and ti2 begin at the same time and ti1 ends before ti2 concludes.

3.2 Rule-Based Component for Behavior Representation

The rule-based component extends the formal representation of states and events provided by the ontology-based component and defines the relationships between them.

The compound term *begin* signifies an event that initiates a state. It is denoted as *begin(state)*, for example, *begin(run)* = *startOfRun*. Additionally, *begin(begin(state), state)*.

The compound term *finish* signifies an event that concludes a state. It is denoted as *finish(state)*, for example, *finish(run)* = *endOfRun*. Moreover, *finishes(finish(state), state)*.

Every state is initiated by an event and concluded by an event. Therefore, from the occurrence of states, we can infer the occurrence of the events associated with them.

An event initiating an interval state occurs at the time point that starts the time interval of that state. An event concluding an interval state occurs at the time point that ends the time interval of that state. The event that initiates an instantaneous state is equal to the event that concludes the state, and it occurs at the time point when the state appears.

1. An event initiating an interval state occurs at the time point that starts the time interval of that state.
2. An event concluding an interval state occurs at the time point that ends the time interval of that state.
3. The event that initiates an instantaneous state is equal to the event that ends the state, and it occurs at the time point when the state appears.

The *eventStartEnv* atom represents the event that initiates the XR environment. Therefore, it can be stated that no event can occur earlier than *eventStartEnv*.

The *eventStopEnv* atom represents the event that concludes the XR environment. Consequently, it can be said that no event can occur later than *eventStopEnv*.

Events signify the beginning and end of states, thereby determining their duration. The duration of a state is the difference between the time point of the event that concludes the state and the time point of the event that initiates the state. It is defined in the domain of time points and denoted by a compound term.

The duration can be utilized to calculate the length of time a particular state lasts in a selected time domain. Since each state is initiated and concluded by an event that determines the duration of the state.

In the case of an instantaneous state, the events that initiate and conclude the state are equal, which means the duration of an instantaneous state is zero. On the other hand, for an interval state, the events that initiate and conclude the state are different. As a result, the duration of an interval state is greater than zero.

Behavior. In practical applications, there is a need for direct relationships between states and the temporal entities in which these states occur. The concepts of event calculus can be used to achieve this goal.

The immediate evaluation predicate (IEP) is true for a given state or event and a timepoint if and only if the state or event occurs at that timepoint. In particular, IEP predicates include *holdsAt* and *time*. *holdsAt(state, tp)* assesses whether a state occurs at a timepoint tp. This predicate can be applied to both instantaneous and interval states, as each state can be evaluated at a timepoint.

To evaluate whether a state is initiated or terminated at a given timepoint, the IEP can be used. This evaluation can be performed using the time predicate along with the *begin* and *finish* terms, which represent events. The predicates *time(begin(state), tp)* and *time(finish(state), tp)* assess whether a state is initiated and terminated, respectively, at a timepoint tp.

Similarly, we define a predicate that evaluates the occurrence of states in time intervals (ETOI). It is true for a given interval state and a time interval if and only if the interval state occurs within the time interval. The ETOI predicate is *holds(state, ti)*.

The temporal evaluation predicate (TEP) is either IEP or ETOI and can be used to evaluate the occurrence of states in timepoints or time intervals.

State Transitions. States and events can trigger other states and events. For instance, the XR environment is active (state) from the moment it begins (event). In a factory, a stamping press initiates operation (state) when the start button is pressed (event) and proceeds to shape the product (state) as a result of contact with the metal material (event). Moreover, states can depend on other states, such as a house being fully illuminated if each room has been illuminated, regardless of the order in which lights were turned on in individual rooms. This allows for the creation of arbitrary cause-and-effect chains described by transitions that are consistent with the event-condition-action model. There are two types of transitions: event-based transitions and state-based transitions. An event-based transition is a Horn clause that consists of:

1. Body, which is a conjunction of a statement based on the IEP predicate that evaluates the occurrence of an event, and any number of statements based on predicates that are not fluent,
2. Head, which is a statement based on the TEP predicate.

The *assert* predicate is a predicate that is true for a given statement if and only if the statement exists in the rule set or it is possible to add the statement to the rule set. The predicate is denoted as *assert(statement)*. In examples, the predicate is often used to associate time points with time intervals in transitions. We assume that the *assert* predicate is satisfied unless stated otherwise.

4 Example

4.1 General Information

The introduced semantic behavior model was used to represent the virtual environment of the industrial worker training XR system, which allows trainees to learn how to act safely in an industrial setting.

The system has been developed in the Unity game engine and uses an Oculus Quest 2 VR headset. Users can interact with objects using their own hands, which was archived by using the Oculus Integration plug-in together with supplementary plug-ins to ensure a high level of immersion.

The training scenario implemented in the system focuses on safe work with the industrial press. It was developed using resources from Amica S.A., a major producer of household equipment in Poland. The prepared virtual scene in which the training takes place is shown in Fig. 2.

The scenario consists of several steps:

1. The worker pulls the metal sheet out of the container
2. The worker visually inspects the physical condition of the metal sheet to be processed looking for flaws
3. The worker places a metal sheet in the press
4. The worker lubricates/sprays the metal sheet parts with oil
5. The worker starts the press using the button
6. The worker takes the finished product from the press

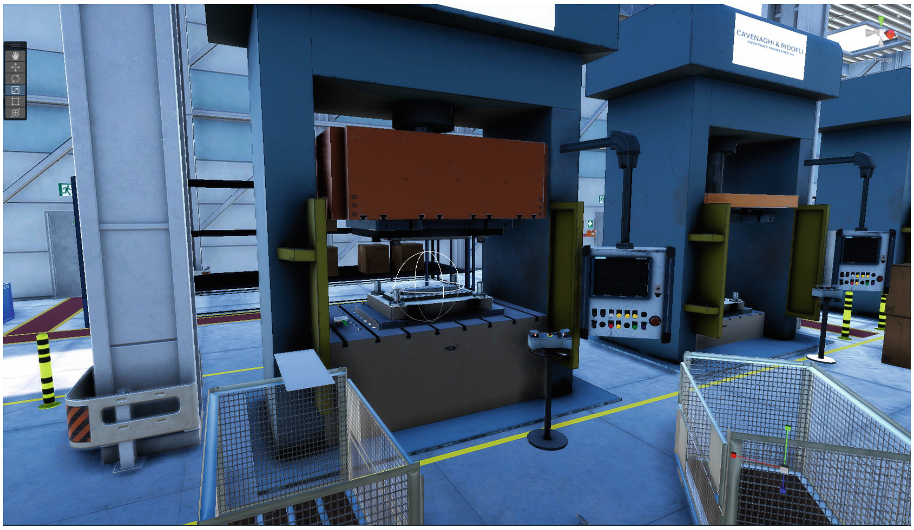


Fig. 2. An industrial press in a factory hall

4.2 Knowledge Base Example

In the scene, there are three main objects which are used in the scenario: the metal sheet, the industrial press and the button for turning on the industrial press. The system generates a knowledge base that describes these objects. The example of such a knowledge base presents Listing 1.1. The objects are described in lines 1–15. Every object has its name, id and possible states; for example, Object2 (lines 6–9) - the industrial press has id: 2, and three possible states: empty press, press with inserted metal sheet and working press. The first two states are instant states, while the last state is an interval state. Additionally, object3 (a press button) is part of the industrial press, which is described in line 13 by object property isObjectOf. In lines 17–47, the rest possible states are described, also by their name and id.

Listing 1.1. A fragment of knowledge base describing objects in the scene and their possible states

```
 1  fo:Object1 rdf:type owl:NamedIndividual , fo:Object ;
 2             fo:hasState fo:SheetState1 , fo:SheetState11 , fo:
                   SheetState2 , fo:SheetState3 , fo:SheetState4 , fo:
                   SheetState5 ;
 3             fo:id "1"^^xsd:int ;
 4             fo:name "metal␣sheet"^^xsd:string .
 5
 6  fo:Object2 rdf:type owl:NamedIndividual , fo:Object ;
 7             fo:hasState fo:PressState10 , fo:PressState6 , fo:
                   PressState7 ;
 8             fo:id "2"^^xsd:int ;
 9             fo:name "industrial␣press"^^xsd:string .
10
11  fo:Object3 rdf:type owl:NamedIndividual , fo:Object ;
12             fo:hasState fo:buttonState8 , fo:buttonState9 ;
```

```
            fo:isObjectOf fo:Object2 ;
            fo:id "3"^^xsd:int ;
            fo:name "press␣button"^^xsd:string .

fo:PressState10 rdf:type owl:NamedIndividual , fo:IntervalState ;
                fo:id "10"^^xsd:int ;
                fo:name "working␣press"^^xsd:string .

fo:PressState6 rdf:type owl:NamedIndividual ,  fo:InstantState ;
               fo:id "6"^^xsd:int ;
               fo:name "empty␣press"^^xsd:string .

fo:PressState7 rdf:type owl:NamedIndividual ,  fo:InstantState ;
               fo:id "7"^^xsd:int ;
               fo:name "press␣with␣inserted␣metal␣sheet"^^xsd:string
                    .

fo:SheetState11 rdf:type owl:NamedIndividual , fo:InstantState ;
                fo:id "11"^^xsd:int ;
                fo:name "trimmed␣sheet"^^xsd:string .

fo:SheetState3 rdf:type owl:NamedIndividual ,  fo:InstantState ;
               fo:id "3"^^xsd:int ;
               fo:name "Visualy␣checked"^^xsd:string .

fo:SheetState4 rdf:type owl:NamedIndividual ,  fo:InstantState ;
               fo:id "4"^^xsd:int ;
               fo:name "metal␣sheet␣inside␣press"^^xsd:string .

fo:buttonState8 rdf:type owl:NamedIndividual , fo:InstantState ;
                fo:id "8"^^xsd:int ;
                fo:name "unpressed"^^xsd:string .

fo:buttonState9 rdf:type owl:NamedIndividual , fo:InstantState ;
                fo:id "9"^^xsd:int ;
                fo:name "pressed"^^xsd:string .
```

Every action that happens in the scene is presented in the knowledge base as an event. Each described event has its own name and informs which states it begins and finishes. Moreover, each event has assigned a time slice, which informs when the event took place.

The Listing 1.2 presents the knowledge base generated by the system when the user inserts a metal sheet into an industrial press (Fig. 3). It generates an event (lines 1–7) that finishes two states: pressState6 (empty press) and SheetState3 (Visually checked), and begins two states pressState7 (press with inserted metal sheet) and SheetState4 (metal sheet inside press). The event has an object TimeSlice assigned (lines 9–12). Since the event happened over a period of time, additionally time interval was assigned (TimeInterval13). Time interval inform when the event started and ended (lines 14–17).

Listing 1.2. Example of knowledge base generated when user inserts metal sheet into press

```
1  fo:Event4 rdf:type owl:NamedIndividual ,
2                     fo:Event ;
3            fo:begins fo:PressState7 ,
4                      fo:SheetState4 ;
5            fo:finishes fo:PressState6 ,
6                        fo:SheetState3 ;
7            fo:name "inserting␣metal␣sheet␣into␣hydraulic␣press"^^xsd:
                 string .
8
```

```
 9  fo:TimeSlice4 rdf:type owl:NamedIndividual ,
10                         fo:TimeSlice ;
11                fo:hasTimeInterval fo:TimeInterval3 ;
12                fo:isTimeSliceOf fo:Event4 .
13
14  fo:TimeInterval3 rdf:type owl:NamedIndividual ,
15                            fo:TimeInterval ;
16                   fo:end "2023-04-26T09:07:52"^^xsd:dateTime ;
17                   fo:start "2023-04-26T09:06:34"^^xsd:dateTime .
```

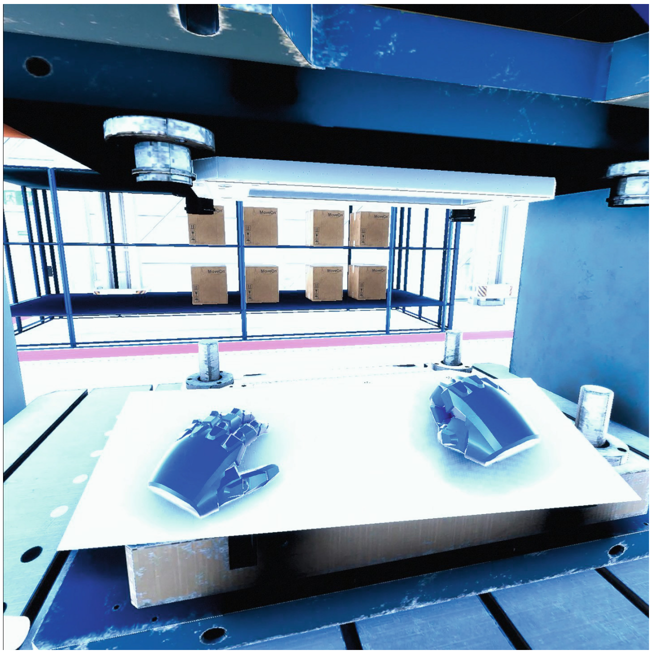


Fig. 3. User inserts metal sheet into industrial press

Another fragment of the generated knowledge base is presented in the Listing 1.3. When the user pushes the button (Fig. 4) to activate the industrial press, the event is generated (lines 1–7). This event finishes two states: pressState7 (press with inserted metal sheet) and ButtonState8 (button unpressed), and begins two states pressState10 (working press) and ButtonState9 (button pressed). Event has assigned the time slice (lines 9–12), which by data property time point informs in what exact time the event happened (line 12). After that, another event (stopping the operation of the press) is generated, which describes when and what happens in the scene when the press stopped working (lines 14–24).

Listing 1.3. Example of knowledge base generated when user pushes the button of the industial press

```
1  fo:Event6 rdf:type owl:NamedIndividual ,
2                     fo:Event ;
```

```
          fo:begins fo:PressState10 ,
                    fo:ButtonState9 ;
          fo:finishes fo:PressState7 ,
                      fo:ButtonState8 ;
          fo:name "pressing␣turn␣on␣button␣of␣the␣press"^^xsd:string
              .

fo:TimeSlice6 rdf:type owl:NamedIndividual ,
                        fo:TimeSlice ;
              fo:isTimeSliceOf fo:Event6 ;
              fo:timePoint "2023-04-26T09:09:02"^^xsd:dateTime .

fo:Event7 rdf:type owl:NamedIndividual ,
                    fo:Event ;
          fo:begins fo:SheetState11 ;
          fo:finishes fo:PressState10 ,
                      fo:SheetState5 ;
          fo:name "stopping␣the␣operation␣of␣the␣press"^^xsd:string .

fo:TimeSlice7 rdf:type owl:NamedIndividual ,
                        fo:TimeSlice ;
              fo:isTimeSliceOf fo:Event7 ;
              fo:timePoint "2023-04-26T09:09:36"^^xsd:dateTime .
```

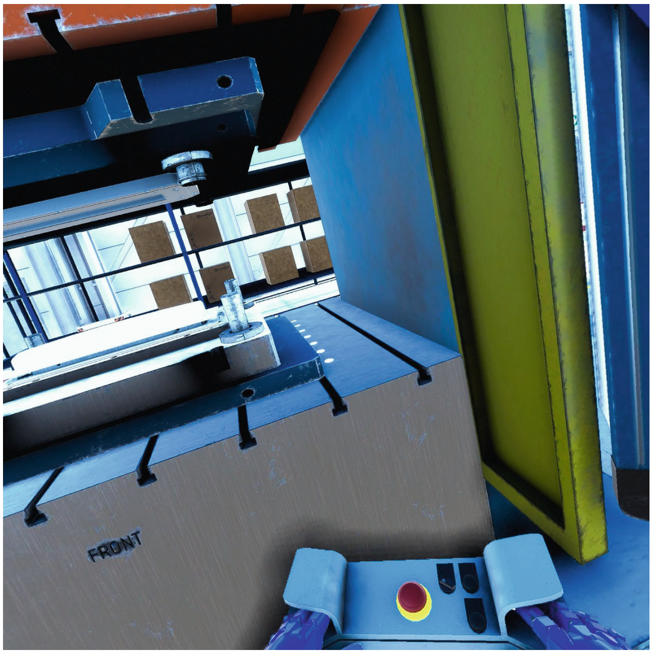


Fig. 4. User pushes the button to activate the press

4.3 Exploration Queries

After generating the knowledge base, it is possible to explore it using semantic queries. It gives the possibility to gain a broad range of information about the progress of training, for example, what events happened during the training process, when they happened, how the states of objects in the scene changed etc.

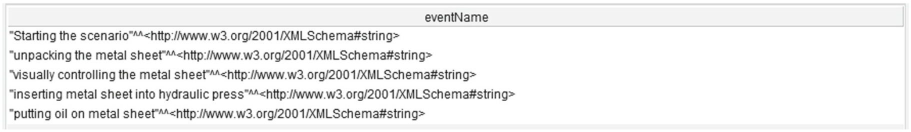
The exploration is possible due to the use of queries encoded using SPARQL, which is the main query language for RDF-based ontologies and knowledge bases. Below are presented examples of such queries and their results:

Query 1. Which events had happened before the press button was pushed?

```
numberstyle
SELECT  ?eventName
WHERE { ?event rdf:type fo:Event  .
            ?timeSlice fo:isTimeSliceOf ?event .
            ?event fo:name ?eventName .
          { ?timeSlice fo:timePoint ?time . }
          UNION
          {
            ?timeSlice fo:hasTimeInterval ?timeInterval .
                ?timeInterval fo:start ?time .
          }

            ?PushButtonEvent fo:name "pressing␣turn␣on␣button␣of␣the␣
                 press"^^<http://www.w3.org/2001/XMLSchema#string> .
            ?PushButtonTimeSlice fo:isTimeSliceOf ?PushButtonEvent .
            ?PushButtonTimeSlice fo:timePoint ?pushButtonTime .

            FILTER (  ?time < ?pushButtonTime)
}
ORDER BY ?time
```



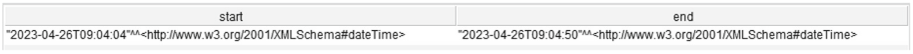
```
eventName
"Starting the scenario"^^<http://www.w3.org/2001/XMLSchema#string>
"unpacking the metal sheet"^^<http://www.w3.org/2001/XMLSchema#string>
"visually controlling the metal sheet"^^<http://www.w3.org/2001/XMLSchema#string>
"inserting metal sheet into hydraulic press"^^<http://www.w3.org/2001/XMLSchema#string>
"putting oil on metal sheet"^^<http://www.w3.org/2001/XMLSchema#string>
```

Fig. 5. Results of query 1

The first query provides information about what happened in the scene before the trainee pushed the press button. The query searches for events and their assigned time slices that happened before the event with the name "pressing the turn on the button of the press". The result consists of event names ordered by time, which is shown in Fig. 5.

Query 2. How long the metal sheet was inspected?

```
numberstyle
SELECT ?start ?end
WHERE { ?event fo:name "visually␣controlling␣the␣metal␣sheet"^^<http
     ://www.w3.org/2001/XMLSchema#string> .
                ?timeSlice fo:isTimeSliceOf ?event .
                ?timeSlice fo:hasTimeInterval ?timeInterval .
                ?timeInterval fo:start ?start .
                ?timeInterval fo:end ?end .
        }
```



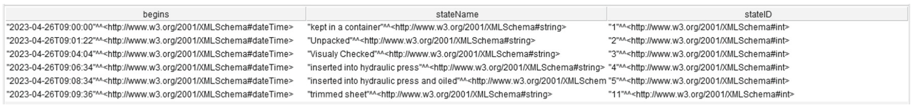
start	end
"2023-04-26T09:04:04"^^<http://www.w3.org/2001/XMLSchema#dateTime>	"2023-04-26T09:04:50"^^<http://www.w3.org/2001/XMLSchema#dateTime>

Fig. 6. Results of query 2

The second query gives information about the duration of the visual inspection of the metal sheet. The query searches for the time slice of the event named "visually controlling the metal sheet", then using the time interval object, access the information when the event started and ended. The result of the query is shown in Fig. 6.

Query 3. When and What states the sheet metal transitioned into?

```
numberstyle
SELECT ?begins  ?stateName ?stateID
WHERE{ ?state rdf:type fo:InstantState .
           ?object fo:hasState ?state .
           ?object fo:name "metal␣sheet"^^xsd:string .
           ?state fo:name ?stateName .
           ?state fo:id ?stateID .
           ?event fo:begins ?state .
           ?timeSlice fo:isTimeSliceOf ?event .
           ?event fo:name ?eventName .
         { ?timeSlice fo:timePoint ?begins . }
         UNION
         {
           ?timeSlice fo:hasTimeInterval ?timeInterval .
           ?timeInterval fo:start ?begins .
         }
        }
ORDER BY ?begins
```



begins	stateName	stateID
"2023-04-26T09:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime>	"kept in a container"^^<http://www.w3.org/2001/XMLSchema#string>	"1"^^<http://www.w3.org/2001/XMLSchema#int>
"2023-04-26T09:01:22"^^<http://www.w3.org/2001/XMLSchema#dateTime>	"Unpacked"^^<http://www.w3.org/2001/XMLSchema#string>	"2"^^<http://www.w3.org/2001/XMLSchema#int>
"2023-04-26T09:04:04"^^<http://www.w3.org/2001/XMLSchema#dateTime>	"Visualy Checked"^^<http://www.w3.org/2001/XMLSchema#string>	"3"^^<http://www.w3.org/2001/XMLSchema#int>
"2023-04-26T09:06:34"^^<http://www.w3.org/2001/XMLSchema#dateTime>	"inserted into hydraulic press"^^<http://www.w3.org/2001/XMLSchema#string>	"4"^^<http://www.w3.org/2001/XMLSchema#int>
"2023-04-26T09:08:34"^^<http://www.w3.org/2001/XMLSchema#dateTime>	"inserted into hydraulic press and oiled"^^<http://www.w3.org/2001/XMLSchem	"5"^^<http://www.w3.org/2001/XMLSchema#int>
"2023-04-26T09:09:36"^^<http://www.w3.org/2001/XMLSchema#dateTime>	"trimmed sheet"^^<http://www.w3.org/2001/XMLSchema#string>	"11"^^<http://www.w3.org/2001/XMLSchema#int>

Fig. 7. Results of query 3

The last query provides information about what happened sequentially with the metal sheet. The query searches for events that changed the states of the scene object named "metal sheet". Then using proper time slices, the query determines the time by which the states are ordered. The results consist of the name and ID of the states and the time when they have begun. The query result is shown in Fig. 7.

Such queries can provide information about user experience with the training system. For example, how many mistakes the user made, how long the interaction with different objects took, and how complex and hard were the activities for the user. This information can be then used to evaluate the system and training scenario in order to increase the functionality and user-friendliness of the system and boost the efficiency of the virtual training.

5 Conclusions and Future Work

The use of knowledge-based representation for interactive 3D content, especially in the context of the collaborative network- and web-based VR/AR systems, has the potential to benefit various application domains by providing additional information about users' and objects' behavior. The paper presents a new approach to semantically representing the behavior of XR environments, including users' and objects' interactions and autonomous actions. The proposed model allows for the representation of temporal characteristics of XR environments with domain-specific terminology, which can be further explored through queries and reasoning. This approach could potentially lead to a more effective and widespread distribution of domain knowledge using XR. Furthermore, the information acquired by semantic queries can improve the user experience by measuring the performance of the users inside the virtual scene, which can lead to evaluating the system and training scenario in order to enhance the usability of the training system.

Additionally, an example and overview of the generated knowledge base for the VR training system have been presented, which is a real training scenario based on the requirements of the training process house appliances manufacturer. Additionally, on generated knowledge base the semantic queries were performed and results and overview were presented.

Possible future research directions can focus on the development of semantic repositories with domain knowledge collected using the proposed model as well as the development of machine learning methods for analyzing and classifying the collected data. It could be especially useful to discover patterns in users' behavior in training sessions, e.g., common problems and mistakes, to improve the overall training performance. In addition, we plan to develop graphical tools supporting the execution of queries against training ontologies and knowledge bases, which would facilitate the use of our approach by non-IT-specialists. Moreover, formal responses to semantic queries can be enriched by the accompanying visualization recorded during the training session.

Acknowledgements. The presented research was co-funded by the National Center for Research and Development under grant number LIDER/55/0287/L-12/20/NCBR/2021.

References

1. Chmielewski, J.: Describing interactivity of 3d content. In: Cellary, W., Walczak, K. (eds.) Interactive 3D Multimedia Content, pp. 195–221. Springer, London (2012). https://doi.org/10.1007/978-1-4471-2497-9_8
2. Chokwitthaya, C., Zhu, Y., Lu, W.: Ontology for experimentation of human-building interactions using virtual reality. Adv. Eng. Inform. **55**, 101903 (2023). https://doi.org/10.1016/j.aei.2023.101903, https://www.sciencedirect.com/science/article/pii/S1474034623000319

3. Chu, Y.L., Li, T.Y.: Realizing semantic virtual environments with ontology and pluggable procedures. In: Applications of Virtual Reality. IntechOpen, Rijeka (2012). https://doi.org/10.5772/36761
4. Chu, Y., Li, T.: Using pluggable procedures and ontology to realize semantic virtual environments 2.0. In: Proceedings of The 7th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry. VRCAI '08, pp. 27:1–27:6. ACM, New York, NY, USA (2008)
5. De Troyer, O., Kleinermann, F., Pellens, B., Bille, W.: Conceptual modeling for virtual reality. In: Grundy, J., Hartmann, S., Laender, A.H.F., Maciaszek, L., Roddick, J.F. (eds.) Tutorials, Posters, Panels and Industrial Contributions at the 26th International Conference on Conceptual Modeling - ER 2007. CRPIT, vol. 83, pp. 3–18. ACS, Auckland, New Zealand (2007)
6. Flotyński, J., Walczak, K.: Ontology-based representation and modelling of synthetic 3D content: a state-of-the-art review. Comput. Graph. Forum, 1–25 (2017). https://doi.org/10.1111/cgf.13083
7. Heitmayer, M., Russell, M.G., Lahlou, S., Pea, R.D.: An ontology for human-centered analysis and design of virtual reality conferencing. In: TMS Proceedings 2021, 3 November 2021. https://tmb.apaopen.org/pub/3rbumwgw
8. Kapahnke, P., Liedtke, P., Nesbigall, S., Warwas, S., Klusch, M.: ISReal: an open platform for semantic-based 3D simulations in the 3D internet. In: International Semantic Web Conference (2), pp. 161–176 (2010)
9. Pellens, B., De Troyer, O., Bille, W., Kleinermann, F.: Conceptual modeling of object behavior in a virtual environment. In: Proceedings of Virtual Concept 2005, pp. 93–94. Springer, Biarritz (2005). https://doi.org/10.1007/2-287-28773-6
10. Pellens, B., De Troyer, O., Bille, W., Kleinermann, F., Romero, R.: An ontology-driven approach for modeling behavior in virtual environments. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM 2005. LNCS, vol. 3762, pp. 1215–1224. Springer, Heidelberg (2005). https://doi.org/10.1007/11575863_145
11. Pellens, B., De Troyer, O., Kleinermann, F.: Codepa: a conceptual design pattern approach to model behavior for x3D worlds. In: Proceedings of the 13th International Symposium on 3D Web Technology, pp. 91–99. Los Angeles, 09–10 August 2008
12. Pellens, B., Kleinermann, F., De Troyer, O.: A development environment using behavior patterns to facilitate building 3D/VR applications. In: Proceedings of the 6th Australasian Conference on International Entertainment. IE '09, pp. 8:1–8:8. ACM (2009)
13. Rabattu, P.Y., et al.: My corporis fabrica embryo: an ontology-based 3D spatio-temporal modeling of human embryo development. J. Biomed. Semant. **6**(1), 36 (2015). https://doi.org/10.1186/s13326-015-0034-0
14. Sikos, L.F.: 3D model indexing in videos for content-based retrieval via X3D-based semantic enrichment and automated reasoning. In: Proceedings of the 22Nd International Conference on 3D Web Technology. Web3D '17, pp. 19:1–19:7. ACM, New York, NY, USA (2017). https://doi.org/10.1145/3055624.3075943, http://doi.acm.org/10.1145/3055624.3075943
15. Sikos, L.F.: Spatiotemporal reasoning for complex video event recognition in content-based video retrieval. In: Hassanien, A.E., Shaalan, K., Gaber, T., Tolba, M.F. (eds.) AISI 2017. AISC, vol. 639, pp. 704–713. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-64861-3_66
16. Trellet, M., Ferey, N., Baaden, M., Bourdot, P.: Interactive visual analytics of molecular data in immersive environments via a semantic definition of the content

and the context. In: 2016 Workshop on Immersive Analytics (IA), pp. 48–53. IEEE (2016)
17. W3C: Sparql query language for RDF (2008). http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/. Accessed 24 Mar 2015
18. W3C: Owl (2015). http://www.w3.org/2001/sw/wiki/OWL. Accessed 24 Mar 2015
19. W3C: Rdf (accessed March 24, 2015), http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/
20. W3C: RDFs (2015). http://www.w3.org/TR/2000/CR-rdf-schema-20000327/. Accessed 24 Mar 2015