VR-WISE: A Conceptual Modeling Approach for Virtual Environments

Bram Pellens, Wesley Bille[1], Olga De Troyer, Frederic Kleinermann
{Bram.Pellens, Wesley.Bille, Olga.DeTroyer, Frederic.Kleinermann}@vub.ac.be
Research Group WISE, Vrije Universiteit Brussel
Pleinlaan 2, 1050 Brussel, Belgium

Abstract. Regardless the growing popularity of Virtual Reality, the design of Virtual Environments remains a complex task that requires skilled people. To overcome this, we have developed the VR-WISE approach that allows specifying a Virtual Environment at a conceptual level, free from any implementation details, allowing stakeholders not familiar with VR-technology to participate in the design of a VR application. In this paper, we illustrate our approach and in particular the modeling concepts by means of an elaborated example of a VR mechanical drilling robot.

1. Introduction

Today, there is a growing interest in 3D technologies due to a continuously improvement of both hard- and software. However, although the creation of Virtual Reality (VR) applications is supported by means of a number of software tools, the development of a Virtual Environment (VE) is still a difficult and tedious task. Software tools for VR development can be divided in either toolkits (like Performer [8]) or authoring tools (like 3D Studio Max [5]). Although they assist the developer in creating a VE, they require considerable background knowledge about VR technology. A novice user, who does not have this background, is therefore excluded from developing VR applications. Furthermore, the design phase in the development process of a VR application (from the perspective of a classical software engineering life cycle) is usually an informal activity. Few formal techniques exist to support the VR design phase effectively. A systematic approach that uses the output of the design phase as input for the implementation does not exist.

To deal with these concerns, we have developed an approach called VR-WISE [2] that aims at facilitating and shortening the development process of VR applications by means of conceptual specifications, also called *Conceptual Models*. Conceptual models are high-level, implementation independent descriptions of the VE that we want to build. For this, VR-WISE provides a number of high-level modeling concepts that can be used to specify the conceptual models. In this way, the developer does not have to deal with low level VR modeling primitives from toolkits or authoring tools, but can express the VE in terms of the application domain and using intuitive, high-level modeling concepts. The actual VE will then be automatically generated from these conceptual models. We believe that such an approach will facilitate VR development for less experienced users and hence allow opening up the discipline to a broader audience than nowadays. For the high-level modeling concepts developed so far, we also provide a graphical notation. Using the graphical notation, different diagrams can be made for the different models needed to specify a VE. A graphical language for expressing designs has the regular advantages over a text-based one: it enhances the communication between designers, programmers and other stakeholders because diagrams are easier in communication; they are more efficient in their use; and

[1] Research Assistant for the Fund for Scientific Research - Flanders

sophisticated tools can be developed that prevent errors and provide views on the design from different perspectives and on different levels of abstraction.

In this paper we provide an overview of the VR-WISE approach and illustrate the modeling process by means of an elaborated example of a drilling robot. The paper is organized as follows. Section 2 discusses some related work on methods for developing VEs. In section 3, we shortly introduce the VR-WISE approach we developed for describing VEs at a conceptual level. The software tool that has been implemented to support this research is briefly discussed in this section as well. Section 4 describes the example that will be used to illustrate our modeling approach. In order to do this we elaborate the drilling robot example in section 5 by describing first how to model the static structure followed by discussing the modeling of the behavior. We end this paper with a conclusion and future work in section 6.

2. Related Work

Developing VEs has never been easy or intuitive for non VR-skilled persons. A number of research groups have developed ways to facilitate the specification of VEs.

The lack of high-level design methodologies for VR development has been addressed in [10] with the presentation of VRID which divides the design process into a high-level design phase and a low-level design phase. Due to the latter phase, this method is not usable for novice users. Another approach which can be related to ours is Ossa [9]. It provides a system for modeling of VR using Conceptual Graphs and logical rules. However, the proposed modeling concepts are rather limited and therefore also the expressiveness of the model. In [4], a software engineering approach is presented to design VEs. The specification is divided into three interrelated aspects: form, function and behavior. It allows specifying the VR application by means of a set of graphical diagrams. Although this approach is suitable for software engineers, it might not be the case for novice users. In VR-WISE we facilitate this by incorporating domain knowledge into the design process. In [6] the advantages of the use of ontologies are also recognized. However, both the contents of the VE and the semantic information contained in the ontologies need to be created separately. As we will see in the following section, using the VR-WISE approach, we automatically obtain the semantic description.

3. VR-WISE Approach

As already explained, the VR-WISE approach provides a way to develop a VE using conceptual specifications (also called *Conceptual Models*). This is done using high-level modeling concepts together with domain terminology. In this way, during the specification, we allow to abstract from the implementation details. In addition, the use of conceptual models may improve the reusability, extensibility and modularity of the VE.

In VR-WISE, the development process is composed of three (mainly) sequential steps, namely the *specification* step, the *mapping* step and the *generation* step. In terms of software engineering, the specification step covers the design phase. It allows the designer to specify the VE at a high level using domain concepts without taking into account any implementation details. In a nutshell, this is done by defining the (domain) objects needed in the VE, their properties, the relationships among them and their behaviors. The mapping step allows specifying mappings from the conceptual level into the implementation level. The purpose of this mapping is to specify how the domain objects should eventually be represented in the VE. The generation step generates the actual source code for the VE. In summary, we can state that the approach provides a systematic way to make a conceptual design and to convert it into a working application. An elaborated overview of the approach can be found in [2] and [3].

To support the approach, we have developed a prototype tool called *OntoWorld*. It allows the designer to intuitively make a conceptual design, specify the mappings and finally generate the code for the VE in an X3D format [11].

4. The Example: A Virtual Drilling Robot

In order to illustrate our approach, we use the example of a mechanical drilling robot (figure 1). The purpose of this paper is to show how the complete VE presented in figure 1 can be specified using VR-WISE. We first describe this VE in an informal way.

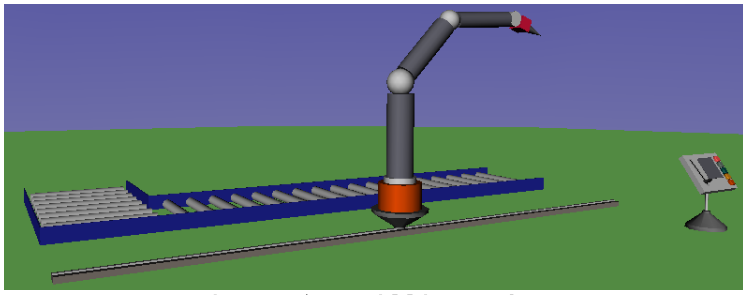

Figure 1. Drilling Robot

The scene consists of a robot together with a console that can be used for manipulating the robot. The robot has a body which can move along a rail. A mechanical arm is placed on top of that body. The arm consists of a number of different arm-sections, which can move independently according to some joints. Next to the arm-sections there is also a drill head. More details will be given while we elaborate the example in section 5.

5. Specifying the Virtual Environment

The modeling approach used by VR-WISE is an Object-Oriented (OO) approach. This means that the main modeling concepts are *concepts* and *instances*. Concepts are comparable with classes (or object types); instances are comparable with class instances (or objects). Concepts are used to model (describe) the relevant concepts of the application domain. The main concept for the robot example is ‘DrillRobot’, which involves parts like ‘Rail’, ‘Body’ ‘ArmSection’, ‘DrillHead’ and ‘Drill’. As in OO, concepts and instances can have attributes, which represent their relevant properties. The Rail concept can for example have a length, depth, width and color property. Instances will inherit the attributes defined for their concept. Instances are used to represent the actual objects in the VE. For example, we can have a VE with one DrillRobot(-instance) or with multiple DrillRobot(-instance)s.

In fact, specifying a VE (in the VR-WISE specification phase) is done at two levels. The first level is called the *domain level* and the second level is called the *world level*. The domain level allows specifying the different types of domain objects (concepts) that may be needed in the VE. At the world level, the VE is composed by specifying the actual objects (e.g. DrillRobot instances) that need to populate the world and by placing them in the VE. The world level also allows specifying general properties of the VE such as gravity, lights, cameras, etc. In this paper, we concentrate on the modeling of the domain level[2]. We

[2] In principle it would be possible to omit the domain level and directly specify the objects itself, but this is currently not supported.

specify the drill-robot at the domain level. Doing so allows reuse of the drill-robot specification in a VE with multiple drill-robots (or in another VE).

In the rest of this section, we discuss how the example can be specified by means of the high-level modeling concepts of VR-WISE. As already indicated, the modeling concepts can be used at the domain level as well as at the world level. At the domain level they are used in combination with concepts, at the world level they are used in combination with instances. In the descriptions given below, we will not distinguish between concepts and instances but we will use the term “object” instead.

5.1 Modeling the static structure of the Drill Robot

In this section we will explain in more detail how to model an articulated body inside the VR-WISE approach by means of the drill robot example.

Concepts - Instances. The first thing a designer needs to do when modeling an articulated body is modeling the different *concepts* that will be part of it. Our Drill Robot involves concepts like a Rail, a Body, a DrillHead and a Drill. These concepts are represented as a rectangle in figure 4. Instances are graphically represented with an ellipse. As already mentioned, we model the Drill Robot on the domain level so no instances are needed here and hence no examples are in figure 4.

Roles. The ArmSection concept is represented (in the middle and lower left part of figure 4) by another graphical notation than a regular concept. This is because it plays three different roles in the model. It plays the role of the lower arm, the middle arm and the upper arm. Therefore the modeling concept *role* is introduced. Concepts and instances may play different roles. A role can be seen as a concept (or instance) together with the role it plays in the context in which it is used. The concept (or instance) itself can be defined inside or outside the context in which its role is used. A role is represented by means of a double-sided rectangle. The name of the role as well as the name of the corresponding concept (or instance) is mentioned. In figure 4 some examples are shown, namely the lower_arm, middle_arm and upper_arm.

Spatial Relations. Next, the designer can specify the position of the objects in the VE. Our approach offers the possibility to express the positioning of objects relative to some other object(s). In our Drill Robot example, the Body is modeled ‘on top’ of the Rail (upper part of figure 4). This high-level and intuitive way for positioning is provided by means of *spatial relations*. The following set of spatial relations is available in our approach: ‘Left’, ‘Right’, ‘Front’, ‘Back’, ‘Top’, ‘Bottom’, ‘Middle’ and ‘OnTop’. Note that these different spatial relations can be combined to form new ones. The spatial relations are graphically represented by means of a rounded rectangle with the type of spatial relation and the distance between the two concepts positioned relative to each other. Inside the rectangle a symbol denotes the type of relation, namely the spatial relation (figure 2).

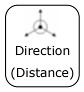

Figure 2: Graphical notation for the Spatial Relations

Orientation Relations. So far, in our example the different parts of our Drill Robot have been modeled and are positioned with respect to each other. However, the concepts also need an orientation. The middle_arm has another orientation than for example the upper_arm. Our approach offers two types of *orientation relations* for this purpose. The first way is to orient two objects with respect to each other. An example can be found between the Rail and the Body object (upper part of figure 4). This orientation relation states that the ‘top’ side of the Rail is oriented towards the ‘bottom’ side of the Body. Different sides can be combined (e.g. the top-left side of object X is oriented towards the left side of object Y). This relation is also represented by means of a rounded rectangle containing a symbol denoting the type of relation, namely the orientation relation (figure 3a).

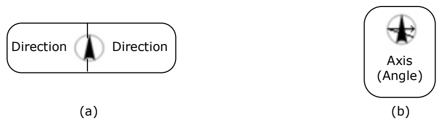

Figure 3: (a) Orientation relation, (b) Orientation-by-angle relation

In the example of the Drill Robot, the middle_arm is not oriented relative to another object. Our approach supports this by means of a second type of orientation relation, namely *orientation-by-angle relation.* Using this type of orientation the middle_arm can be oriented by rotating it 30 degrees over its front-to-back axis (figure 3b). This relation is not placed between two objects but is just attached to one single object, the one that needs to be given the orientation. It can be seen in the middle of figure 4 on the left where an orientation-by-angle relation on the middle_arm role allows rotating it 30 degrees over its front-to-back axis.

Connection Relations. So far we have defined the different concepts that make part of our Drill Robot example, they are positioned relative to each other and they have also been given correct orientations. For the Drill Robot to form one whole, the different objects (e.g. body, the arm sections, the drill-head) need to be connected to each other.

In our approach three *Connection Relations* are provided. First, we have the *Connection Point Relation* that connects two objects over a center-of-motion. A connection over a center-of-motion means that there have to be two points, one on each connected object, which have to fall together at all times in the VE. In our Drill Robot example, such a connection is defined between the lower_arm and the middle_arm. The second one, the *Connection Axis Relation,* connects two objects over an axis-of-motion meaning that there has to be two axes, one on each connected objects, which fall together at all times in the VE. Again, in our Drill Robot example, such a connection is defined between the Rail and the Body. The third relation, the *Connection Surface Relation,* connects two objects over a surface-of-motion stating that a surface on both objects has to be on the same position at all times in the VE. In our Drill Robot no connection of this kind is specified at the moment.

We will now discuss one of the relations in more detail, namely the Connection Axis Relation (figure 5b). The Connection Axis Relation is represented by a rounded rectangle containing the connection axis symbol. The rectangle has two parts that are used to define an axis on both objects. The axis is defined as follows. In our approach, three planes through each object are defined, the horizontal plane (defined by the front-to-back and the left-to-right axes), the vertical plane (defined by the front-to-back and top-to-bottom axes) and the perpendicular plane (defined by the left-to-right and the top-to-bottom axes). A connection axis is defined by the intersection of two planes. The designer can translate and rotate the planes using the description PlaneName(Direction, Distance, Rotation). Direction and Distance define the translation; the Rotation defines the rotation angle of the plane. Each plane can only rotate over one axis, the horizontal plane over the left-to-right axis, the perpendicular over the top-to-bottom axis and the vertical over the front-to-back axis. For example, the description “Horizontal(top, 0.5m, 0°)” which is used in the connection axis relation between the Rail and the Body (upper part of figure 4) means that the horizontal plane is being translated upwards for 0.5m and being rotated for 0°. When no arguments are given, the default planes are used without any translation or rotation. An example is shown in figure 4 between the DrillHead and Drill concept with the description “Vertical()”.

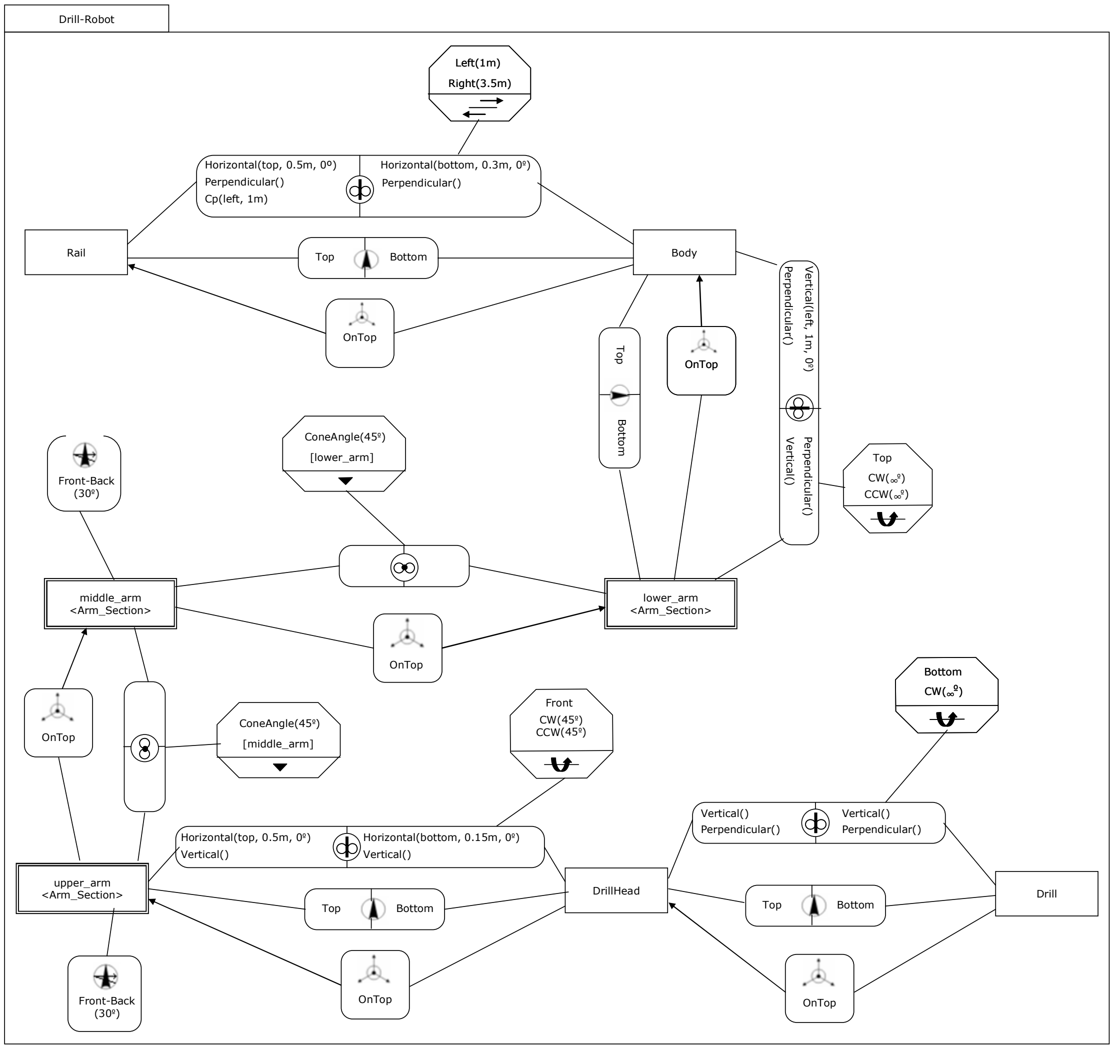

Figure 4: Conceptual Model of the Drill Robot Concept

Both concepts are connected over an axis going through their center so for each object the axis is defined by the intersection of the vertical and the perpendicular plane. Note that there is no need for a translation or rotation in this example.

The remaining connection relations, the Connection Point relation (figure 5a) and the Connection Surface (figure 5c), will not be discussed in detail here.

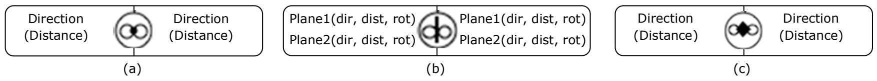

Figure 5: Graphical notation of the Connection Relations

Connection Constraints. Different relations that can be used to connect two objects to each other have now been discussed. The difference in the connection relations will influence the behavior of the objects. So far, the behavior on the connections would be unlimited. In order to be able to restrict the behavior, some constraints need to be specified. For example, the Connection Axis Relation between the Body concept and the Rail concept only states that we have a connection over an axis-of-motion but not which restrictions are set on the motion.

Therefore our approach contains *connection constraints*. In order to allow the Body and the Rail to move along the axis, a *prismatic constraint* is placed on top of the connection axis relation. This constraint forces the bodies connected by means of the connection axis to move along the axis and not to rotate around it. It is only necessary to specify the direction and the distance of the maximal movement. In our example the Body (because the constraint is connected on that side of the relation) is specified to allow a movement of 1 meter to the left and 3.5 meters to the right over the connection axis. This can be found in figure 4 at the top.

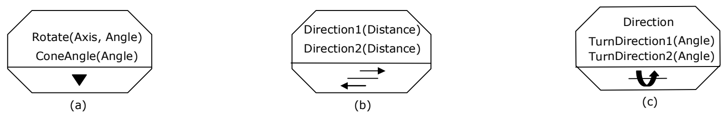

Figure 6: Graphical notation for the Prismatic Constraint

The graphical notation for the connection constraints is a hexagon together with a symbol as shown in figure 6b. Other constraints available in our approach are the angular limit constraint (figure 6a) and the hinge constraint (figure 6c) that will not be explained in detail due to space limitations. A hinge constraint is defined on the connection between the DrillHead and the Drill (figure 4 on the bottom right). Angular limit constraints are defined on the arm sections (figure 4 in the middle).

5.2 Modeling the behavior of the Drill Robot

In this section, we will now go into more detail on the modeling of the behavior by means of the VR-WISE approach using and extending on the same example of the Drill Robot. In general, the behavior specification process is divided into two steps: definition specification and invocation specification. A more detailed and complete overview is given in [7].

A behavior definition allows the designer to define different behaviors for an object. The behaviors of an object are defined separately from its static structure and independent of how the behavior will be triggered. The complete behavior definition is given in figure 9.

Actors. The main modeling concept in a behavior definition is an *actor*. An actor represents an object that is involved in a behavior. Because we separate the definition of a behavior from the definition of the structure of an object, actors are used in the definition of a behavior instead of the actual object(s). An actor is thus a kind of abstract object. An actor is graphically represented by a circle with the name of the actor written inside. Examples of actors in figure 9 are Lever, RobotBody, Actuator, etc. For an actor, only the minimal properties needed to obtain the desired behavior should be specified (not shown graphically here). This implies that each object that has those minimal properties can replace the actor and thus have the associated behavior.

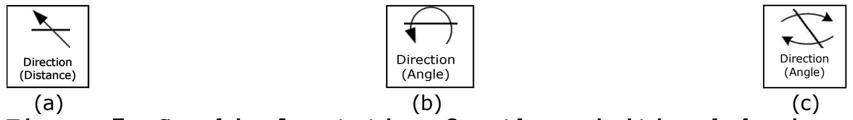

Figure 7. Graphical notation for the primitive behaviors

Behavior. A behavior can be defined for every actor. The focus of this paper is on a particular type of behavior describing the movement of objects. We distinguish between primitive behavior and composite behavior. Composite behaviors can be defined by combining behaviors (either primitive or composite ones) using operators. A behavior is graphically represented by means of a rectangle. For primitive behavior, the rectangle carries a symbol denoting the type of primitive behavior as well as some additional information (i.e.

parameters). In the context of our Drill Robot example, a number of primitive behaviors have been provided for describing behavior for the connections described in previous section.

The *move-over-axis* (figure 7a) is used to denote the movement of an object along a particular axis. In order to specify this action, a direction and a distance needs to be given. The direction has a value which can be either ‘left’, ‘right’, ‘front’, ‘back’, ‘top’, ‘bottom’ or any valid combination of these. Only the directions that are given in the constraint, as seen in section 5.1, can be used as a value for the direction here. The distance is given by means of a real value or a variable, and a unit. See figure 9b where the RobotBody of the robot is moved for a certain distance along the rail using this kind of action.

The *turn-over-axis* (figure 7b) is used to express a rotate of the object along an axis that is defined by a connection axis relation together with a hinge constraint. For this action, a direction and an angle are needed. The direction can in this case only be ‘clockwise’ or ‘counter-clockwise’ as was specified in section 5.1. The angle expresses how much the object needs to be turned and is given by a value (or variable) and a unit. In figure 9a this is illustrated by the LowerArm that is turned over the axis for 90°.

The *roll-over-axis* (figure 7c) describes a rotate over a connection point relation together with a cone limit constraint. The action also requires a direction and an angle to be given. The direction can be one (or multiple) of ‘left’, ‘right’, ‘front’, ‘back’, ‘top’, ‘bottom’ depending on the values given in the constraint. The angle is given by means of a value (or variable) and a unit. An example can be seen in figure 9c where the MiddleArm is rolled over de axis of motion for 15 degrees. As can be seen on figure 9, the axis along which the movement is being performed is denoted by means of a reference-link.

Remaining primitives are: *move*, *turn*, *roll*, *position* and *orient*. These either change the position of an object or its orientation. Only the roll is used in this example (figure 9b).

Operators. Composite behavior can be achieved by combining other behaviors (either primitive or composite) by means of operators. We can use *temporal operators* (to synchronize the behaviors, figure 8a) [1], *lifetime operators* (in order to control the lifetime of a behavior, figure 8b) and *conditional operator* (to control the flow of the behavior on the basis of a condition, figure 8c). They are represented by a rounded rectangle with a symbol denoting the kind of operator. Additional information is written below the symbol.

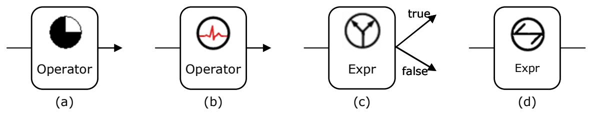

Figure 8. Graphical notations for the Temporal(a), Lifetime(b), Conditional(c), Influential(d) operator

In addition to these operators we have also defined an *influential operator* (figure 8d). The influential operator indicates the inter-relationships of the movements of the objects involved in the relation; it tells us how the movement of one object can influence the movement of another object and vice versa. Inside this element, the coupling between the actions to which it is connected is described (figure 9c on the right where the movement of MiddleArm results in the movement of the UpperArm and the Head). The operator can have a direction (given by the arrow on the relation) meaning that the influence is not mutual but only valid in one way. The use of the influential operator is extremely useful when modeling mechanical devices e.g. gears, belts and pulleys,…

The second step involves creating *behavior invocations* in which the behaviors defined are assigned to the actual objects (concepts or instances) and the designer specifies how the behaviors can be triggered. This separation between the definition of the behavior and the specification of how the behavior can be triggered improves reusability and enhances flexibility since the same behavior definition can be reused for different objects (if different types of objects have the same behavior) and the same behavior can be triggered in different ways (e.g. by some user interaction or by a collision with another object). Here, the robot and its parts that were modeled in section 5.1 are being assigned to the defined behavior.

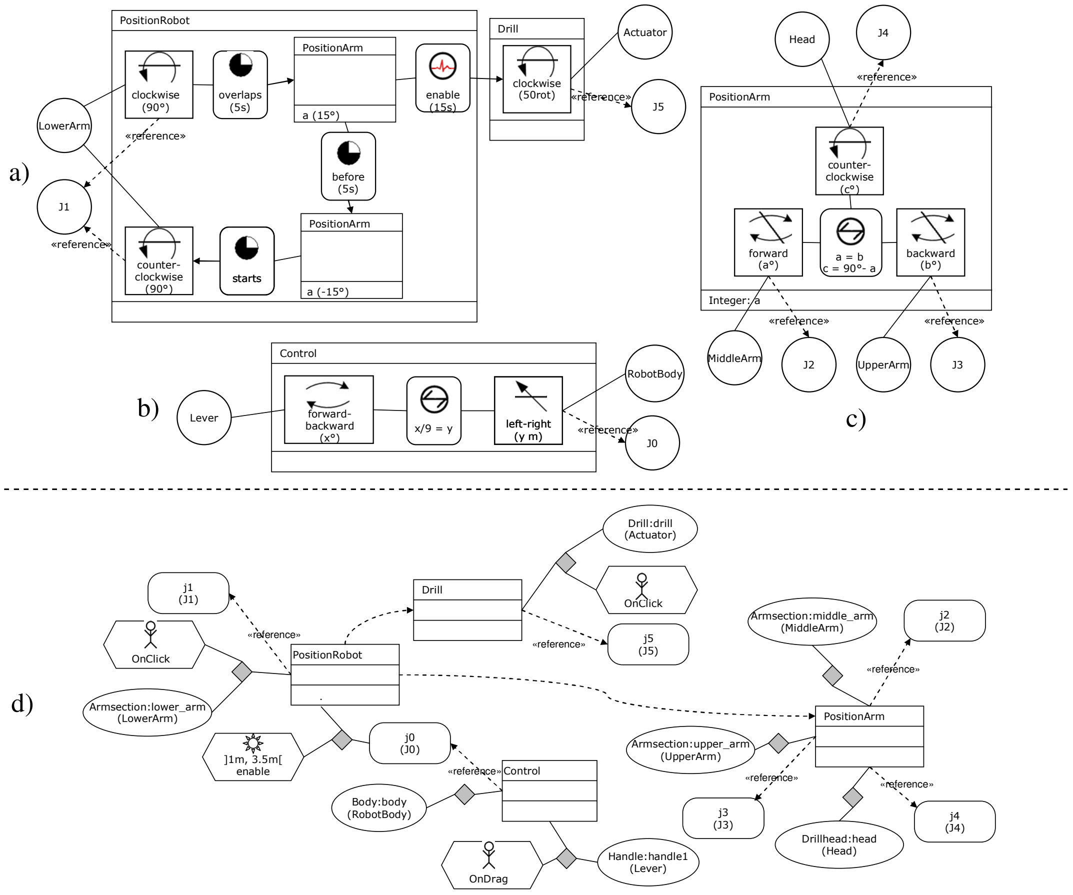

Figure 9. (a)(b)(c) Behavior Definition, (d) Behavior Invocation

Concepts – Instances. Concepts and instances defined in the static structure can get (an) actor(s) assigned. In our example, the Actuator actor has been assigned to an instance of the Drill concept (figure 9d). By assigning an actor to a concept, we couple behavior to the concept, i.e. every instance of that concept will have all the behaviors defined for the actor. By assigning an actor to an instance, only that particular instance will have all the behaviors of the actor. Concepts as well as instances can have multiple actors being assigned. The graphical notation is a bit different from the ones in section 5.1 since here also the actor that is assigned is denoted within brackets under the name of the concept or instance.

Events. In our approach, behaviors are triggered by means of *events*. Events are graphically represented by a hexagon with a symbol denoting the type of event and some additional information below the symbol. In the context of the Drill Robot example a *constraint-event* (figure 10d) is provided which allows us to react when a particular constraint has been reached or, in the worst case, been broken. The constraint-event takes a position or a set of positions, given by an interval and an action which is taken. Using this constraint, we can allow taking action in intermediate positions of the objects in the joint as well. In figure 9, the PositionRobot behavior will be enabled when the joint is in one of the positions as is given by the interval. We additionally have three other kinds of events: a *timeEvent* (figure 10a), a *userEvent* (figure 10b) and an *objectEvent* (figure 10c). These will not be discussed in detail here.

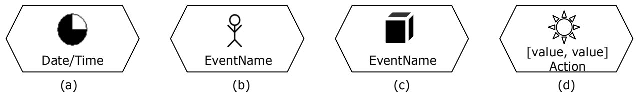

Figure 10. Graphical notation for Events

6. Conclusions and Future Work

In this paper, we have explained the VR-WISE approach by means of an elaborated example. VR-WISE provides a way to describe VEs at a high level, using conceptual models. To achieve this, a VE can be expressed in terms of domain objects. The modeling concepts presented, allow expressing articulated bodies by connecting different rigid bodies using spatial relations and different types of joints. Different constraints can be expressed on the joins to limit the movement of the connected objects with respect to each other. Dedicated modeling concepts are used for modeling behavior. The modeling of behavior is done in an action-oriented way and independent from the objects on the one hand and from the interaction used to invoke the behavior on the other hand, thereby improving the reusability. We also illustrated the graphical notation used for the different modeling concepts. The combination of a conceptual design phase and the use of an intuitive graphical language can drastically reduce the complexity of building dynamic and interactive Virtual Environments.

Future work will focus on user experiments in order to evaluate the usability of our approach, the intuitiveness of the modeling concepts and the graphical notation. We will also extend the current approach with more high-level modeling concepts to enable the specification of more complex structures as well as more advanced physical behavior.

References

[1] Allen, J. F., "Maintaining Knowledge about Temporal Intervals". Communications of the ACM, vol. 26, no. 11, pp. 832-843, 1983

[2] Bille, W., Pellens, B., Kleinermann, F. and De Troyer, O., "Intelligent Modelling of Virtual Worlds Using Domain Ontologies", In Proceedings of the Workshop of Intelligent Computing (WIC), Mexico City, Mexico, 2004, pp. 272-279

[3] De Troyer, O., Bille, W., Romero R. and Stuer P., "On Generating Virtual Worlds from Domain Ontologies", In Proceedings of the 9th International Conference on Multimedia Modeling, Taipei, Taiwan, 2003, pp. 279-294

[4] Kim, G.J., Kang, K.C., Kim, H. and Lee, J., "Software Engineering of Virtual Worlds", In Proceedings of the ACM Symposium on Virtual Reality Software & Technology, Taipei, Taiwan, 1998, pp. 131-139

[5] Murdock, K.L., "3DS Max 5 Bible", Wiley Publishing Incorporated, 2003

[6] Otto, K., "Semantic Virtual Environments", In: the proceedings of the 14th international World Wide Web conference, ACM press, Chiba, Japan, 2005, pp. 1036-1037

[7] Pellens, B., De Troyer, O., Bille, W. and Kleinermann, F., "Conceptual Modeling of Object Behavior in a Virtual Environment", In Proceedings of Virtual Concept 2005, Biarritz, France, 2005 (accepted)

[8] Rohlf, J. and Helman, J., "IRIS Performer: A High Performance Multiprocessing Toolkit for Real--Time 3D Graphics", In Proceedings of the 21st annual conference on Computer Graphics and Interactive Techniques, Orlando, Florida, USA, 1994, pp. 381-395

[9] Southey, F. and Linders, J., "Ossa - A Conceptual Modelling System for Virtual Realities", In Proceedings of the International Conference on Conceptual Structures, Stanford, USA, 2001, pp. 333-345

[10] Tanriverdi, V. and Jacob, R.J.K., "VRID: A Design Model and Methodology for Developing Virtual Reality Interfaces", In Proceedings of ACM Symposium on VRST. Alberta, Canada, 2001, pp. 175-182

[11] Web 3D Consortium (Web3D), Extensible 3D (X3D) International Standard, 2003, http://www.web3d.org/x3d/specifications/