Proceedings of the 2012 Winter Simulation Conference
C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A.M. Uhrmacher, eds

SIMURENA − A WEB PORTAL FOR OPEN EDUCATIONAL SIMULATION

Gerd Wagner

Brandenburg University of Technology
Institute of Informatics
P. O. Box 101344
03013 Cottbus, GERMANY

ABSTRACT

In this paper, we first discuss the question why simulation is still not widely used in education today. We identify three inhibitors and four facilitators for the use of simulation in education. In particular, we point out that educational simulations should be created and distributed as *Open Educational Resources*. Second, we report on the innovative Simurena Portal (Simurena 2012), which is the first simulation portal that satisfies all four requirements for facilitating the use of simulation in education.

1 INTRODUCTION

For a number of reasons, simulation is still not widely used in education today. The three most important reasons are:

1. Making simulations requires a huge effort and is, therefore, expensive, especially if they contain multimedia contents.
2. A lot of educational simulations are closed commercial software products. Average schools or universities can often not afford to buy such products. It is also not possible to adapt these commercial simulations for reuse in different educational settings.
3. Simulations that are free and open source can often not be used in an educational setting because they have not been designed for educational use, so their installation requires too much effort and they do not run in the web browser and on tablets.

We believe that for attaining a more widespread use of simulation in education it is essential to

1. support the creation, publication and reuse of simulation models as *Open Educational Resources*, as discussed in Section 2;
2. allow turning simulations into games by adding a user interface for *participatory simulation*, and by adding motivational elements, as discussed in Section 3;
3. exploit the multimedia capabilities of *HTML5*, and turn simulations into true web resources, as discussed in Section 4;
4. support tablet computers and other *mobile devices*, since these platforms will be the most important ones in education.

In Section 5, we describe the Simurena Portal and in Section 6 the Simurena simulation language. Finally, in Section 7 we compare Simurena with other simulation portals.

2 OPEN EDUCATION

UNESCO defines *Open Educational Resources (OER)* as teaching, learning or research materials that are in the public domain or released with an intellectual property license that allows for free use, adaptation,

and distribution (UNESCO 2012). The idea of OER dates back to the pioneering move of the Massachusetts Institute of Technology (MIT) in 2001 to publish its course materials on the web and allow their free use (MIT 2012).

Figure 1: UNESCO's OER logo.

The OER proposal has also been endorsed by the National Science Foundation (NSF), which recommends in a recent report (NSF 2008) to ´adopt programs and policies to promote *Open Educational Resources*'.

The *Open Education (OE)* movement aims to make educational materials as well as teaching and learning tools, freely available via the world-wide web. This includes tools that support the creation, publication and reuse of simulation models as OER.

For supporting their world-wide *reuse* as OER, simulation models

- must be open source, e.g. by assigning a suitable *Creative Commons* license allowing derivative works;
- should be true *web resources*, that is, they should be written using web standards such as the UTF-8 character encoding, Uniform Resource Identifiers, XML and Dublin Core metadata.

While we may consider OER as 'socialized' digital assets, another goal of the OE movement is to develop and provide 'socialized' open tools and systems for creating and publishing OER, including their adaptation for reuse. So, this goal calls for establishing suitable web portals and for the formation of online communities of OER developers. In the case of simulation models and simulations, this means that an OE simulation platform should provide a web portal that supports the collaborative online development of simulation models and simulations, and the formation of an online community of simulation developers.

For being broadly accessible in today's media technology landscape, OE resources should exploit the multimedia capabilities of *HTML5* and support tablet computers and other *mobile devices*. This implies that simulation models are implemented as JavaScript programs that can be directly executed in the web browser, providing seamless integration with HTML5 and support of mobile platforms. This issue is further discussed in Section 4.

3 PARTICIPATORY SIMULATION AND SIMULATION GAMES

For supporting the use of simulations in education, it is essential that a given simulation model can be 'gamified', that is, turned into a game by adding 1) a suitable user interface allowing the user to participate in the simulation, and 2) suitable motivational elements. Ordinary simulations provide limited opportunities for user interaction. Only simulation games have the potential of being immersive and supporting active learning.

Participatory simulation is a form of multi-agent simulation (MAS) that allows the user to participate in a simulation run by controlling (or 'playing') one of the agents defined in the simulation scenario. This means that for supporting participatory simulations, a simulation technology must allow adding an ***agent control user interface*** to a given simulation model such that the resulting participatory simulation scenario has the same runs as the underlying MAS model, provided the user does not choose to participate in the runs. A participatory simulation model can thus be viewed as a conservative extension of a MAS model according to the following symbolic equation:

participatory simulation model = MAS model + participation model + agent control UI

In summary, we obtain the following symbolic equation for defining an agent control user interface:

agent control UI = output panels + user action forms + user action event listeners

We further discuss these general simulation concepts in the following subsections. We also show how to implement a simulation scenario for participatory simulation in Simurena in Section 5.

3.1 Example

For being able to illustrate our concepts we use the MIT Beer Game model published in (Simurena 2012), which is version of the classical simulation game by Sterman (1992), as an example of a MAS model that can be extended by adding a participation model and a corresponding agent control UI. This model is about a beer supply chain consisting of four nodes: the retailer, the wholesaler, the distributor and the factory. Every intermediate node has one upstream node to order and receive beer from and one downstream node to receive orders from and to delivery beer to.

An agent-based implementation of this conceptual model can be obtained by defining

1. an agent type for the bottom supply chain node (the end customer);
2. an agent type for intermediary supply chain nodes (the retailer, the wholesaler and the distributor);
3. an agent type for the top supply chain node (the factory);
4. and the reactive behavior of these supply chain nodes based on a set of rules.

This model can be turned into a participatory simulation model by adding a participation model and a corresponding agent control UI that allows the user to choose one of the intermediary supply chain nodes and control it during a simulation run.

3.2 Participation Model

A participation model for a MAS model specifies four things:

1. Which agents defined in the MAS scenario may be controlled.
2. Which types of in-world actions of the controllable agents the user may perform.
3. Whether user actions are blocking (synchronous) or non-blocking (asynchronous).
4. Which automated behaviors of the controllable agents are to be suspended.

In our Beer Game example, we would allow the user to play any of the three intermediary supply chain nodes. The only type of in-world action the user may perform on behalf of the controlled agent would be to order a quantity of beer. We would choose user actions to be blocking (synchronous), that is, the simulator would wait for the beer order user input in each round. All behavior rules of the controlled agent that are responsible for making beer order decisions would be suspended in favor of the corresponding user action.

3.3 Output Panels

Output panels can be used for displaying information about the current simulation state. They are supposed to provide the information needed by the user for making action decisions. This includes displaying the current values of attributes of the controlled agent or of other in-world objects, and of global variables as well as statistics variables.

In our Beer Game example, we could have an output panel (e.g. on the left side of the screen) displaying the current values of the inventory, the backorder quantity and the accumulated costs of the controlled intermediary supply chain node.

3.4 User Action Forms

A user action form allows the user to enter values for the parameters of a user action and to perform the action by submitting the form (by clicking, or otherwise activating, a form submission button). By associating the form with an action rule of an agent type in the form definition, form submission events are mapped to corresponding actions of the controlled agent.

For instance, for the participatory Beer Game model we might define a user action form with one form field for allowing the user to enter a value for the parameter *quantity* of the action *SendOrder* associated with the form.

3.5 User Action Event Listeners

User action event listeners allow mapping user interface events of a certain type (such as mouse-click or key-press) to in-world action events of a corresponding type (without user parameters) as defined in the simulation model.

In our participatory example Beer Game model we do not need any user action event listener, since the only in-world user action is the ordering of beer which is handled by the user action form described in the previous subsection.

3.6 Simulation Games

A simulation game can be obtained from a participatory simulation model according to the following symbolic equation:

simulation game = participatory simulation model + motivation

which suggests that a game can be developed incrementally by first defining a multi-agent simulation model, then adding a participation model and an agent control UI, and finally adding motivational elements in the form of goals, tasks and rewards for the player.

3.7 Related Work

In (Narayanan and Kidambi 2011), it is pointed out that historically the first forms of user-interactive simulations with visualization, called *visual interactive simulations*, have been developed in the 1980s and 1990s in parallel with the rise of graphical user interface (GUI) technologies. We would like to remark that it is possible to specify a user-interaction model also for a simulation without a GUI, having just a text-based user interface.

Narayanan and Kidambi (2011) maintain that engineering an interactive simulation is different from engineering a non-interactive simulation since the specification of interaction is concurrent with the model specification. As we will show, this is not necessarily true and is, in fact, undesirable. Rather, it is preferable to take an incremental development approach where the user-interaction model (and, in fact, any other part of the overall user interface) is specified as an incremental and separable extension component for a given simulation model.

As can be seen from the table of contents of (Rothrock and Narayanan 2011), and from a Google search for "interactive simulation" yielding 28.000 search results, there are many approaches to, and many implementations of, interactive simulation in all kinds of problem domains. However, there are no proposals for a general conceptual framework capturing the logical concepts of human computer interaction and related user interface elements in simulation independently of an implementation technology..

4 USING HTML5 AND OTHER WEB STANDARDS IN SIMULATION

We briefly discuss two issues: sharing educational simulation models on the web, and running educational simulations as web resources.

For allowing the world-wide reuse of simulation models as OER, they must be true *web resources*, that is, they must be written with the help of web standards. This implies that

- the text of the model must use the character encoding UTF-8 for allowing all kinds of international characters (e.g., in the user interface text items);
- simulation models should be expressed in XML for facilitating their internationalization, publication and reuse;
- the Dublin Core metadata standard should be used for expressing documentation elements such as creator, creation date, etc.

The easiest way of running a computer program, and, in fact, the only one that is acceptable in education for its ease of use, is to run the program in a web browser by invoking a web hyperlink (URL) without requiring the installation of any third-party plug-in (such as Adobe Flash). This also holds for simulation programs. It implies that the program is delivered as a JavaScript program from a web server, using all the multimedia capabilities of HTML5 (in particular, 3D graphics rendered with WebGL).

The Simurena technology is pioneering HTML5/JavaScript-based simulation. Other simulation technologies providing simulations on the web, such as AnyLogic, Forio or NetLogo, are still based on technologies such as Java applets and Flash, which are getting obsolete in the near future, since they are not supported by mobile web browsers.

5 THE SIMURENA PORTAL

The Simurena Portal is an ongoing project for facilitating the development and publication of simulation models and simulations for open education. It was developed by Mircea Diaconescu, Christian Noack, Jens Werner and Gerd Wagner in 2010 at the Brandenburg University of Technology, Germany. It consists of two site views:

- The Simurena Library for the public use of simulations and games by teachers, students and others.
- The Simurena Development Portal for simulation and game developers.

While there are already a number of efforts to provide simulations as educational resources on a web portal, Simurena is the first portal that supports the development of simulations both as *Open Educational Resources* and as true web resources. The Simurena portal is also pioneering simulation engineering for Open Education by providing a simulation development portal with free developer accounts. This portal provides development tools and facilitates the collaborative development and publication of OE simulations.

5.1 The Simurena Library

The simulations and games from the *Simurena Library* can be run for free from the Simurena website (see the screenshot in Fig. 2). They typically have a Creative Commons license with the permission to inspect, modify and reuse their source code. Developers can set up a free personal account for the *Simurena Development Portal*.

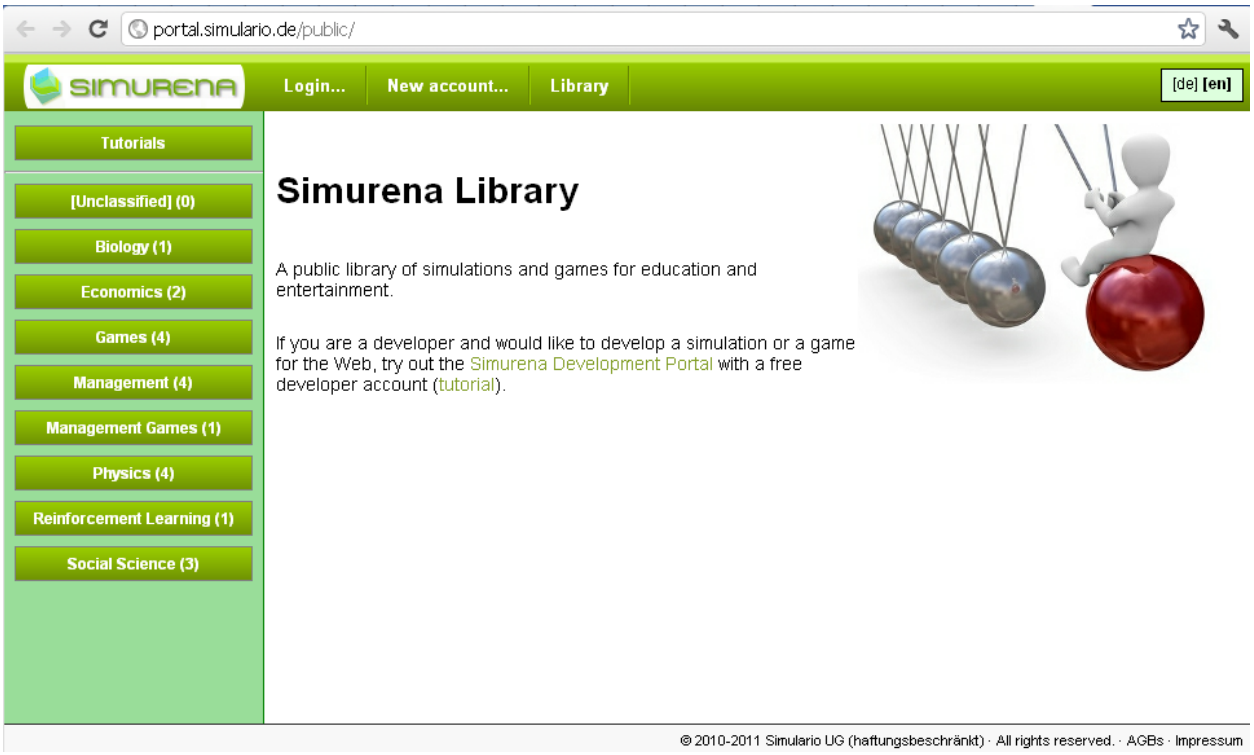

Figure 2: The entry page of the Simurena Library

As can be seen in Fig. 2, the *Simurena Library* classifies simulation models with the help of categories such as *Biology*, *Economics*, *Management*, and *Social Sciences*. E.g., in the *Management* category there are the following models:

- ***Single Queue***: A simple model of a service desk queuing system.
- ***Inventory Management***: A simple inventory management system with a replenishment policy based on a reorder level.
- ***Drive Thru***: A simple model of a drive thru as a system with three order processing units: the order window, the kitchen and the pickup window, and two queues: one for the order window and one for the pickup window.

In the *Economics* category there are the following models:

- ***The MIT Beer Game***: This model describes the classical MIT Beer Game, which is a management simulation with a beer supply chain consisting of five nodes: the end customer, the retailer, the wholesaler, the distributor and the brewery. Every intermediate node has one upstream node to order and receive beer from and one downstream node to receive orders from and to delivery beer to. An order takes 7 days and a delivery takes 14 days. At the end of a week every node decides how much beer to order and calculates its costs for current stock and outstanding orders.
- ***Regional Labor Markets***: Simulates an economy with three goods: labor, an investment (or capital) good and a consumption good, and four types of agents: households, consumption goods producers, malls and capital goods producers, and three markets: an investment goods market, a consumption goods market and a labor market.

5.2 The Simurena Development Portal

The *Simurena Development Portal* allows a developer to maintain a set of simulation scenarios (see the screenshot in Fig. 3) and make decisions if they are to be published in the *Simurena Library* using a

Creative Commons license. For each simulation scenario, the developer may upload one or more associated media files (such as diagrams, images, or sound/music). The portal also allows to check the syntactic correctness of a simulation scenario and to generate the executable JavaScript code. For the future it is planned to provide visual editors that ease the creation of simulation models.

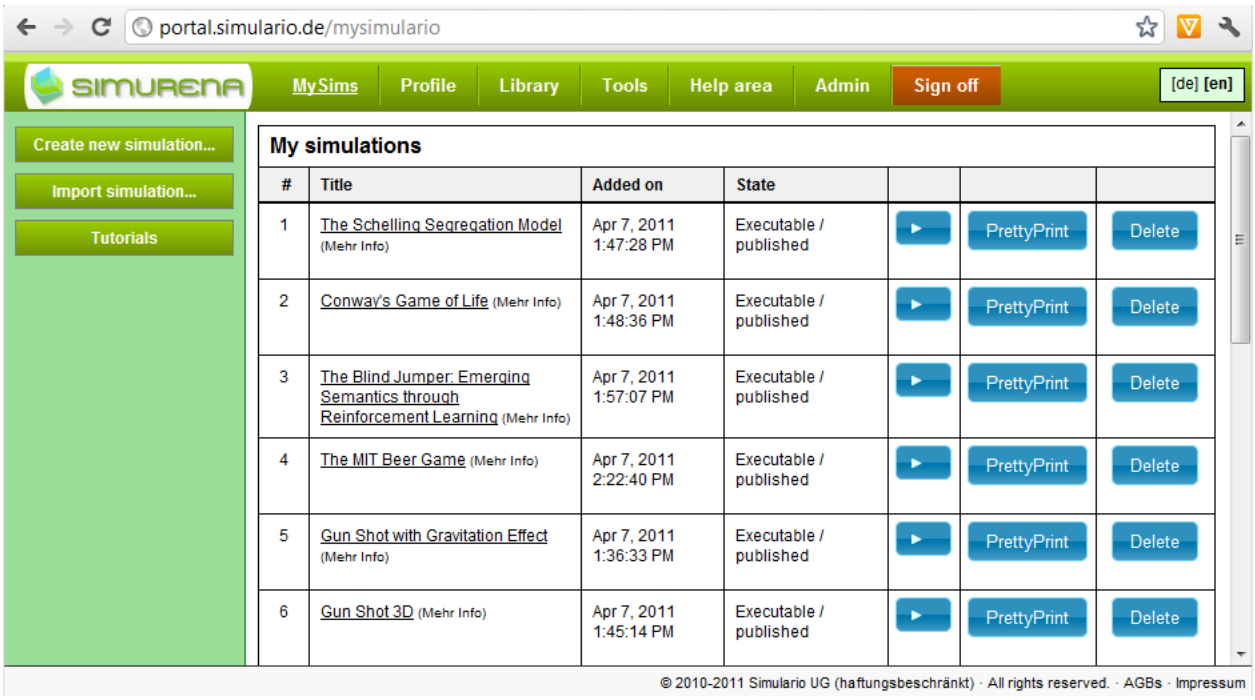

Figure 3: The start page of the Simurena Development Portal

In particular, educational simulations and games, such as management games, can be developed with the *Simurena Portal* in a model-driven manner and then published on the Web. In model-driven simulation engineering, developers can define a user interface model, in addition to the simulation model. Through using the newest web technologies (such as HTML5 and WebGL) the Simurena Portal can generate state-of-the-art user interfaces without requiring the developer to know these technologies.

6 THE SIMULATION LANGUAGE

A simulation model is an executable specification that is either written in a general-purpose programming language or in a special programming language called *simulation language*. The Simurena Portal is using the XML-based *Agent-Object-Relationship* (AOR) simulation language *AORSL* (AORSL 2012, Wagner et al 2009). This language has an XML syntax that allows embedding JavaScript code snippets such as expressions and functions. The XML syntax of AORSL requires some learning effort, but it frees the developer from having to be an expert in JavaScript and Web user interface programming. Only the simulation-specific computations have to be encoded with JavaScript expressions and functions, but not the general simulation logic and environment (such as the user interface).

The following sections discuss the basic concepts of the AOR simulation language.

6.1 Simulation Scenarios and Simulation Models

A simulation scenario consists of

1. simulation parameter settings, such as setting a value for the number of `simulationSteps`,
2. a *simulation model*,

3. an *initial state* definition, and
4. an optional *user interface* definition, including an *initial state user interface*, a *statistics user interface* and/or an *animation user interface*.

A *simulation scenario* consists of

1. a set of entity type definitions, including different categories of event, message, object and agent types; and
2. a set of environment rules, which define causality laws governing the environment's state changes and the causation of follow-up events.

A *simulation model* consists of:

1. a set of *entity type* definitions, including different categories of event, message, object and agent types; and
2. a set of *environment rules*, which define causality laws governing the environment's state changes and the causation of follow-up events.

So, an empty template for a simulation scenario has the following structure:

```
<SimulationScenario xmlns="http://aor-simulation.org"
    version="0.9"
    scenarioName="MySimulationScenario"
    ... >
  <SimulationParameters simulationSteps="1000" ... />

  <SimulationModel modelName="MySimulationModel">
    <documentation> ... </documentation>
    <EntityTypes> ... </EntityTypes>
    <EnvironmentRules> ... </EnvironmentRules>
  </SimulationModel>

  <InitialState> ... </InitialState>

  <UserInterface>
    <InitialStateUI> ... </InitialStateUI>
    <StatisticsUI> ... </StatisticsUI>
    <AnimationUI> ... </AnimationUI>
  </UserInterface>
</SimulationScenario>
```

6.2 Objects and Events

Objects and events are the most fundamental entities of discrete event simulation. Thus, for making a simulation model, suitable object types and event types have to be defined. As an example, we define an object type for a service desk with the attributes `busy` and `queueLength`, and with a function `randomServiceTime`:

```
<ObjectType name="ServiceDesk">
  <Attribute name="busy" type="Boolean"/>
  <Attribute name="queueLength" type="Integer"/>
  <Function name="randomServiceTime" resultType="Integer">
   ...
  </Function>
</ObjectType>
```

There is a distinction between two kinds of events: those that are caused by other events, and those that happen spontaneously (exogenous events). Here is an example of a type definition for caused events:

```
<CausedEventType name="CustomerDeparture">
  <Attribute name="serviceTime" type="Integer"/>
</CausedEventType>
```

In this model of a service desk queuing system, any customer departure (or service end) event is caused by the preceeding service start event. However, our model also contains an example of an exogenous event type, for which a random occurrence periodicity is defined:

```
<ExogenousEventType name="CustomerArrival">
  <Periodicity>
    <DiscreteRandomVariable>
      <UniformInt lowerBound="1" upperBound="8" />
    </DiscreteRandomVariable>
  </Periodicity>
</ExogenousEventType>
```

Notice that the periodicity of `CustomerArrival` events is defined in terms of a random variable based on the uniform distribution of integers between 1 and 8.

Global variables, object types and event types allow defining the state structure of a discrete event system. So we still need to define the dynamics of such a system. That is, for any event type, we have to specify the state changes of affected objects and the follow-up events caused by the occurrence of an event of that type. This can be done with the help of rules, as described in the next section.

6.3 Transition Rules

Transition rules allow defining the dynamics (that is, *state changes* and *follow-up events*) caused by event occurrences. There are two kinds of transition rules in AORSL. Those rules that define the dynamics of the objective environment due to physical causation (the "behavior" of objects) are called *environment rules*. Rules defining the reactive behavior of agents are called *reaction rules*.

Transition rules consist of six elements:

1. `WHEN` specifies the type of event that triggers the rule;
2. `FOR` allows declaring variables, which are either bound to a specific value or object, or to a set of objects;
3. `DO` is an unconditional execution element specifying state changes and follow-up events;
4. `IF` is a logical condition formula possibly containing variables;
5. `THEN` and `ELSE` are two conditional execution elements specifying further state changes and follow-up events.

Any of the execution elements `DO`, `THEN` and `ELSE` may consist of two subelements:

- `UPDATE-ENV`, an expression specifying an update of the environment (or system) state;
- `SCHEDULE-EVT`, an expression specifying a list of follow-up events that will be scheduled.

6.4 Agents

While objects are typically passive entities, agents are *interactive*: they may interact both with their inanimate environment (via perceptions and state-changing actions) and with each other (via message-based communication). As a modeling concept agents can be used for making higher-level simulation models whenever the problem domain includes interacting objects such as biological agents (animals or humans), social agents such as organizations or households, or artificial agents such as robots, machines, or interactive software systems. Since an agent is just a special kind of object, there may be no need to model a given problem with agents. However, modeling interacting objects as agents generally leads to more natural and more readable simulation models.

For modeling agents, we have to define suitable agent types in the simulation model, and then define specific agents of those types in the initial state of the simulation scenario. Like an object type, an agent type also allows defining a set of properties and functions that apply to all instances of the agent type. In addition, it allows specifying:

- reaction rules for defining an agent's reactive behavior (including its communicative behavior);
- internal event types, including periodic time event types and reminder event types;

- a memory size: how many events can be memorized (that is, held in a memory storage).

Reaction rules have a similar syntax like environment rules. They are triggered by perception events, including incoming message events, or by internal time events.

6.5 Animation

Animation is particularly important for educational simulations and games. Simurena simulations can be animated by *visualizing objects* and their state, by *sonifying events* and by allowing human users to *interact* with the simulated world. An *animation user interface* can be added to any simulation model as an incremental extension that does not affect the simulation model.

Objects and their states are visualized by defining a ***view*** for an object type (or for a specific object). A view is defined by a *2D shape* (Square , Rectangle, Triangle, Arc, Circle, Ellipse, RegularPolygon, Polygon, Polyline) or a *3D shape* (Cube, Cuboid, Cone, Cylinder, Pyramid, RegularTriangularPrism, Sphere, Tetrahedra, Mesh).

Events can be *'sonified'* by defining an ***event appearance*** for an event type. This allows attaching specific sounds and melodies to the occurrence of events of that type.

Human users (or players) can interact with the simulated world by performing in-world actions via the user interface. Any simulation model can be turned into an interactive simulation by adding a *participation model* and a corresponding *agent control user interface*.

6.6 Participatory Simulation

The Simurena simulation language AORSL allows adding a *participation model* and a corresponding user interface to a given multi-agent simulation model such that the result is a *participatory simulation*, which can be turned into a *simulation game* by adding motivational elements such as defining a goal that the player has to achieve.

6.6.1 Participation Model

In the AOR simulation model of the Beer Game the participation model is defined in the following way:

```
<AgentControlUI waitForUserInput="true">
  <AgentControlByAgentType type="IntermediarySupplyChainNode"
      suspendReactionRules="EndOfWeek_OrderingBeer_R1_Rule
                            EndOfWeek_OrderingBeer_R2_Rule
                            EndOfWeek_OrderingBeer_R3_Rule
                            EndOfWeek_OrderingBeer_R4_Rule">
  …
  </AgentControlByAgentType>
</AgentControlUI>
```

The element `AgentControlByAgentType` specifies the agent control user interface for the agent type `IntermediarySupplyChainNode`, and implicitly defines all agents of that type to be controllable. It also suspends four rules that together define the ordering behavior of intermediary supply chain nodes. The user-performable in-world actions are implicitly specified by the user action forms and user action event listeners defined in the agent control UI. Since the attribute `waitForUserInput` is set to true, the user action of ordering beer is blocking, that is, the simulator waits in each round for the user to make a beer ordering decision.

6.6.2 Output Panels

Output panels, which allow displaying information about the current simulation state, are part of the agent control UI. An example of an output panel defined in the AOR simulation model of the Beer Game is the following:

```
<LeftOutputPanel>
```

```
  <OutputFieldGroup label="Your state">
    <OutputField label="Inventory" agentAttribute="inventory">
      <Hint>
        <Text>How many crates of beer are in stock?</Text>
      </Hint>
      <Format decimalPlaces="0"><PackagingUnits>cases</PackagingUnits></Format>
    </OutputField>
    <OutputField label="Backorders" agentAttribute="backorderQuantity">
      <Hint>
        <Text>How many crates of beer are in backlog?</Text>
      </Hint>
      <Format decimalPlaces="0"><PackagingUnits>cases</PackagingUnits></Format>
    </OutputField>
    <OutputField label="Cost" agentAttribute="costs">
      <Hint>
        <Text>Your current costs.</Text>
      </Hint>
      <Format decimalPlaces="2"><Currency>EUR(€)</Currency></Format>
    </OutputField>
  </OutputFieldGroup>
</LeftOutputPanel>
```

6.6.3 User Action Forms

For the participatory Beer Game model we might define a user action form with one form field for allowing the user to enter a value for the parameter *quantity* of the action *SendOrder* associated with the form.

In the AOR simulation scenario, this user action form would be defined in the following way:

```
<UserActionForm actionRule="SendOrder" enabledWhenEventOfType="EndOfWeek"
    label="Order">
  <ActionRuleParameterUI parameter="quantity" label="Order quantity">
    <Hint>
      <Text>How many crates of beer do you want to order?</Text>
    </Hint>
    <Format decimalPlaces="0"><PackagingUnits>cases</PackagingUnits></Format>
  </ActionRuleParameterUI>
</UserActionForm>
```

7 OTHER SIMULATION PORTALS

We briefly discuss two other educational simulation portals: MERLOT (2012) and NetLogo (2012), as well as the open source simulation portal OpenABM (2012). All three have severe limitations with respect to accessibility, reuse and creation of simulation models.

MERLOT is an educational resource repository designed primarily for faculty and students of higher education. Although its idea is to support OE, it has many usability problems and allows contributors to upload resources in all kinds of formats and licenses, including non-open resources. Consequently most of its resources are not broadly accessible, since they are not made with web standards. Since many of MERLOT's resources are not open source, reuse is limited. MERLOT does not support the creation of OE resources.

NetLogo is a user-friendly programming language, which is mainly used for making simulation models. Its website provides a large collection of open source simulation models that can be used in OE. However, NetLogo simulation models are not broadly accessible and have only limited reusability, since they are not made with web standards, but with the obsolete Java applet technology.

OpenABM provides various resources for agent-based modeling and simulation, including a repository of open source simulation models, which can be in any format, implying limited accessibility and limited reusability. OpenABM does not support the creation of simulation models.

REFERENCES

AORSL (Agent-Object-Relationship Simulation Language) Reference Manual. 2012. Accessed May 23, 2012. https://oxygen.informatik.tu-cottbus.de/aors/AORSL.html.

MERLOT (Multimedia Educational Resources for Learning and Online Teaching). 2012. Accessed May 23, 2012. http://www.merlot.org/merlot/index.htm

NetLogo. 2012. Accessed May 23, 2012. http://ccl.northwestern.edu/netlogo/

MIT (Massachusetts Institute of Technology) OpenCourseWare. 2012. http://ocw.mit.edu/index.htm

NSF (National Science Foundation). 2008. Fostering Learning in the Networked World: The Cyberlearning Opportunity and Challenge. NSF document number nsf08204. Accessed May 23, 2012. http://www.nsf.gov/publications/pub_summ.jsp?ods_key=nsf08204.

Narayanan, S., and P. Kidambi. 2011. Interactive Simulations: History, Features and Trends. In *Human-in-the-Loop Simulations: Methods and Practice*. Edited by Rothrock, L., and S. Narayanan Springer-Verlag.

OpenABM. 2012. Accessed May 23, 2012. http://www.openabm.org/

Rothrock, L., and S. Narayanan (eds.). 2011. *Human-in-the-Loop Simulations: Methods and Practice*. Springer-Verlag.

Simurena Library. 2012. A public library of simulations and games for education and entertainment. Accessed May 23, 2012. http://portal.simulario.de/public/.

Sterman J.D. 1992. Teaching Takes Off − Flight Simulators for Management Education − "The Beer Game". OR/MS Today, October 1992, 40-44.

UNESCO. 2012. Open Educational Resources. http://www.unesco.org/new/en/communication-and-information/access-to-knowledge/open-educational-resources/. Accessed May 23, 2012.

Wagner G., O. Nicolae and J. Werner. 2009. Extending Discrete Event Simulation by Adding an Activity Concept for Business Process Modeling and Simulation. In: Proceedings of the 2009 Winter Simulation Conference, edited by M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls, December, 13-16, 2009. pp. 2951–2962. Austin, Texas, USA.

AUTHOR BIOGRAPHY

GERD WAGNER is Professor of Internet Technology at the Department of Informatics, Brandenburg University of Technology, Germany. His research interests include agent-oriented modeling and agent-based simulation, foundational ontologies, (business) rule technologies and the Semantic Web. In recent years, he has been focusing his research on the development of an agent-based discrete event simulation framework, called *ER/AOR Simulation* (see www.AOR-Simulation.org). He can be reached at http://www.informatik.tu-cottbus.de/~gwagner/.