Proceedings of the 2013 Winter Simulation Conference
R. Pasupathy, S.-H. Kim, A. Tolk, R. Hill, and M. E. Kuhl, eds.

EXPLORATORY AND PARTICIPATORY SIMULATION

Gerd Wagner

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

ABSTRACT

We discuss two forms of user-interactive simulation: in *exploratory* simulation users may explore a system by means of interventions, and in *participatory* simulation they may participate in a multi-agent simulation scenario by controlling (or 'playing') one of the agents. Exploratory simulation can be used by researchers for validating a simulation model and it can be used by students and trainees for learning the dynamics of a system by interacting with a simulation model of it. Participatory simulation allows dealing with simulation problems where one (or more) of the involved human roles cannot be modeled sufficiently faithfully and therefore have to be played by human actors that participate in simulation runs. We elaborate the concepts of exploratory and participatory simulation on a general, implementation-independent level. We also show how they can be implemented with the AOR Simulation (AORS 2012) platform based on the human-computer interaction paradigm of *agent control*.

1 INTRODUCTION

Simulations involving interactions with human users are often called *human-in-the-loop* simulations, see, e.g., (Narayanan and Kidambi 2011). Sometimes, however, this term is used for the specific simulation category of simulation-based training systems, such as flight simulators, as, e.g., in (Balci, Arthur and Ormsby 2011).

Clearly, user-interactive simulations play an increasingly important role in education and entertainment, but they are also useful for research and design. In this paper, we discuss two forms of user-interactive simulation: *exploratory* simulation allowing users to explore a system by intervening in simulation runs, and *participatory* simulation allowing users to participate in a multi-agent simulation scenario by controlling (or 'playing') one of the agents.

We propose a number of basic logical (implementation-independent) concepts underlying any form of exploratory and participatory simulation and then show how they can be expressed in the XML-based *Agent-Object-Relationship (AOR)* simulation language AORSL. However, we would like to stress that the proposed conceptual framework is independent of the AOR simulation platform and could also be implemented with other simulation platforms.

2 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.

Another term that is used for user-interactive simulation is *Human-in-the-Loop* simulation, which is often associated with Flight Simulators, Driving Simulators, Marine Simulators and the like. In Downes et al (2004), a case of integrating a human-in-the-loop simulation with an agent based model is investigated, however without proposing any general simulation engineering concepts.

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 framework capturing the logical concepts of human computer interaction and related user interface elements in simulation engineering independently of a specific implementation technology.

3 EXPLORATORY SIMULATION

Exploratory simulation allows a human user to explore a system (or, more precisely, a simulation model of a system) by intervening in a simulation run according to ***intervention rules*** defined in an *intervention model* as an extension of a given simulation model. This means that for supporting exploratory simulation, a simulation technology must allow adding an intervention model, and a corresponding intervention user interface (UI), to a given simulation model such that the resulting exploratory simulation model has the same runs as its underlying simulation model, provided the user does not make use of her intervention possibilities. An exploratory simulation model can thus be viewed as a conservative extension of a simulation model according to the following symbolic equation:

exploratory simulation model = simulation model + intervention model + intervention UI

In exploratory simulation, a user action can be understood at two levels of abstraction. At the physical level, a user action is a *user interface event* brought about by a human user with the help of a user input device (such as a keyboard, a mouse or a touch screen). At the logical level, a user action can be interpreted as an ***intervention action*** in the context of an intervention model. *User action event listeners*, discussed below, map user actions at the physical UI level to corresponding intervention actions.

Intervention rules are functions that map intervention actions to *simulation events* or to *simulation state changes*, in the context of a given simulation state. In summary, we obtain the following symbolic equation for defining an intervention model:

intervention model = intervention actions + intervention rules

There are two kinds of user actions: those having parameters, such as *move(x,y)* for moving to a new location given by the coordinates x and y, and those having no parameters, such as *moveNorth* for moving in a grid space to the neighbor cell with coordinates x and y+1, when the current coordinates are x and y. User actions with parameters may in general be implemented as form submission events with suitable form fields in the UI, while parameter-free user actions may be implemented as simple UI events (such as keyboard events or mouse click or touch events).

An exploratory simulator must therefore allow 1) defining ***user action forms*** for user actions with parameters, and 2) mapping form submission events and simple UI events to corresponding intervention actions. While form submission events can be mapped to corresponding intervention actions by associating the form with an intervention action event type, a mapping of simple UI events to corresponding intervention actions can be defined in the form of ***user action event listeners***, which are similar to the *event listeners* of the W3C Document Object Model (DOM3 2011), associating an intervention action event type with a simple UI event type.

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

intervention UI = user action forms + user action event listeners

We will say more about these concepts, and also show how they are supported in AOR simulation, in the following subsections.

3.1 Example

For being able to illustrate our concepts we now introduce an example of a simulation model that will be extended by adding an intervention model. We use the Segregation Model of Schelling (1971), which is about residential neighborhoods with different kinds of residents and their decisions to move or not to move to another neighborhood based on their degree of intolerance towards neighbors of a different kind. A simple implementation of this conceptual model can be obtained by defining a cellular-automata-style discrete event model where

1. residential locations are represented by the cells of a grid space;
2. residents of different kinds are represented by different values of a suitable grid cell property (such as *kindOfResident*);
3. and the behavior of residents consists of periodically making (and implementing) movement decisions at each time step (representing a discrete time event).

The main research question to be investigated with the help of the model is: do residents segregate even if they are quite tolerant towards their neighborhood including many residents of a different kind? The answer to this question is *yes, they segregate even if they are quite tolerant*, as already found by Schelling (1971) with the help of his original non-computerized simulation model, and confirmed many times later with the help of various computerized simulation models, see e.g. (Gilbert 2002).

This model can be turned into an exploratory simulation model by adding an intervention model that allows the user to move a particular resident to a new location after a segregation equilibrium has been reached, and then observe the reactions of the other residents in his new neighborhood and the formation of a new equilibrium state.

3.2 Intervention Actions

An intervention action is an action performed by the user for exploring a simulation model. It is natural to consider the user as a special agent that participates in the simulation, therefore. This approach requires that the simulation technology used allows modeling the user as an agent whose behavior is defined by a set of *action event types*, representing types of intervention actions.

For instance, for turning our segregation simulation model described in the previous section into an exploratory simulation model, we might define just one type of intervention action: *move(* x_1, y_1, x_2, y_2*)* for moving a resident from location (x_1, y_1) to a free location (x_2, y_2) on the two-dimensional grid.

In the AOR simulation language, such an intervention action would be defined with the help of an *action event type* in the following way:

```
<ActionEventType name="Move">
  <Attribute name="x1" type="Integer"/>
  <Attribute name="y1" type="Integer"/>
  <Attribute name="x2" type="Integer"/>
  <Attribute name="y2" type="Integer"/>
</ActionEventType>
```

3.3 Intervention Rules

Intervention rules are transition functions that map intervention actions to simulation state changes and/or to resulting simulation events, in the context of a given simulation system state. They have

1. a ***triggering event*** type (the type of intervention action),
2. zero or more rule variable declarations,

3. an optional ***condition*** on the simulation state,
4. an expression specifying a ***change of the simulation state***,
5. and an expression specifying one or more ***resulting events***.

For instance, for our exploratory segregation model we might define an intervention rule that maps intervention actions of the form *move(x_1, y_1, x_2 ,y_2)* to corresponding state changes where the affected resident on location (x_1 ,y_1) has moved to the new location (x_2 ,y_2).

In the AOR simulation language, such an intervention rule would be expressed as an *environment rule* in the following way:

```
01 <EnvironmentRule name="Move_InterventionRule">
02   <WHEN eventType="Move" eventVariable="evt"/>
03   <IF> Global.isFreeLocation(evt.x2, evt.y2) </IF>
04   <DO>
05     <UPDATE-ENV>
06       <UpdateGridCell>
07         <XCoordinate> evt.x1 </XCoordinate>
08         <YCoordinate> evt.y1 </YCoordinate>
09         <Slot property="kindOfResident" value="-1"/>
10       </UpdateGridCell>
11       <UpdateGridCell>
12         <XCoordinate> evt.x2 </XCoordinate>
13         <YCoordinate> evt.y2 </YCoordinate>
14         <Slot property="kindOfResident" value="3"></Slot>
15       </UpdateGridCell>
16     </UPDATE-ENV>
17   </DO>
18 </EnvironmentRule>
```

The rule checks for any specific move action event, if the location given by the rule parameters x_2 and y_2 is free, using the global function *isFreeLocation*, and then updates the grid by freeing the location (x_1 ,y_1) and marking the location (x_2 ,y_2) with the presence of a resident of kind 3.

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 through clicking (or otherwise activating) a form submission button. By associating the form with an intervention rule in the form definition, form submission events are mapped to corresponding intervention actions.

For instance, for our exploratory segregation model we might define a user action form with two form fields for allowing the user to enter values for the two parameters x and y of the intervention action *move(x,y)* associated with the form via the corresponding intervention rule.

In the AORS model, this user action form would be defined in the following way:

```
<UserActionForm actionEventType="Move" label="Move">
  <ActionEventParameterUI parameter="x1" label="From X-coordinate"/>
  <ActionEventParameterUI parameter="y1" label="From Y-coordinate"/>
  <ActionEventParameterUI parameter="x2" label="To X-coordinate"/>
  <ActionEventParameterUI parameter="y2" label="To Y-coordinate"/>
</UserActionForm>
```

3.5 User Action Event Listeners

A user action event listener is a function that maps UI events of a certain type to action events of a corresponding type defined in the simulation model, including intervention action types in the case of an exploratory simulation model.

For instance, for our exploratory segregation model we might define a user action event listener that maps click events where the user has clicked on a grid cell to intervention actions.

3.6 Use Cases

There are two obvious use cases for exploratory simulation: 1) it allows a form of interactive validation of simulation models, and 2) in teaching and training by allowing students and trainees to learn the dynamics of a system by directly interacting with a simulation model of it.

4 PARTICIPATORY SIMULATION

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 simulation 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 simulation 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 will say more about these concepts, and also show how to define an AOR simulation model for participatory simulation, in the following subsections.

4.1 Example

For being able to illustrate our concepts we use the MIT Beer Game model published in (Simurena 2012) 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 simple 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.

4.2 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 automated beer order decisions would be suspended in favor of the corresponding user action.

For instance, 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>
```

4.3 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 and statistics variables as well as information about current events.

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. 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>
  <OutputFieldGroup label="System state">
    <OutputField label="Total costs" agentAttribute="costs">
      <Hint><Text>Which costs have been accumulated meanwhile?</Text></Hint>
      <Format decimalPlaces="2"><Currency>EUR(€)</Currency></Format>
      <Source><StatisticsVariable name="systemCosts"/></Source>
    </OutputField>
  </OutputFieldGroup>
</LeftOutputPanel>
```

4.4 User Action Forms

As explained in section 3.4, user action forms allow the user to enter values for the parameters of a user action and to submit 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 in-world action *SendOrder* associated with the form via a corresponding action rule.

In the AORS model, 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>
```

4.5 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. However, we may want to support key press events involving the numeric keys of a keyboard for entering order quantities by defining a suitable user action event listener.

4.6 Use Cases

We mention two use cases for participatory simulation: 1) for dealing with simulation problems where one (or more) of the involved human roles cannot be modeled sufficiently faithfully and therefore have to be played by human actors that participate in simulation runs, and 2) for implementing simulation games.

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

$$\text{simulation game} = \text{participatory simulation model} + \text{motivation}$$

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.

5 CONCLUSION

Despite the considerable interest in, and publications about, forms of user-interactive simulation, often referred to as *Human-in-the-Loop* simulation, no proposals for a general framework for user-interactive simulation have been made. As a first step towards such a general framework for user-interactive simulation we propose two concepts that capture important forms of user-interactive simulation: namely *exploratory* simulation and *participatory* simulation.

ACKNOWLEDGMENTS

The author is grateful to the Centre for Research in Social Simulation at the University of Surrey, UK, for providing a visiting fellowship that has triggered my work on the present paper.

REFERENCES

AORS (AOR Simulation). 2012. Accessed April 16. http://oxygen.informatik.tu-cottbus.de/aor/.

Balci, O., J. D. Arthur and W. F. Ormsby. 2011. Achieving reusability and composability with a simulation conceptual model. *Journal of Simulation*, 1–9.

DOM3 (Document Object Model Level 3 Events Specification). 2011. Accessed April 16. http://www.w3.org/TR/DOM-Level-3-Events.

Gilbert, N. 2002. Varieties Of Emergence. *Proceedings of the Agent 2002 Conference on Social Agents: Ecology, Exchange, and Evolution*, Chicago, IL (USA), October 11-12. Accessed April 16, 2012 http://www.ipd.anl.gov/anlpubs/2003/04/46046.pdf.

Downes, M., J. Kwinn, and D. E. Brown. 2004. Using Agent-Based Modeling And Human-In-The-Loop Simulation To Analyze Army Acquisition Programs. *In Proceedings of the 2004 Winter Simulation Conference*. Edited by R .G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, 994-1000. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

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

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

Simurena. 2012. Accessed April 16. http://portal.simulario.de/public/.

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/. His email is g.wagner@tu-cottbus.de.