An Agent-Based Data-Generation Tool for Situation-Aware Systems

Davide Merico and Roberto Bisiani
NOMADIS Lab., University of Milan-Bicocca, viale Sarca, 336/14, I-20126 Milan MI, Italy.
e-mail: {davide.merico, roberto.bisiani}@nomadis.unimib.it

Abstract — This paper describes the design, implementation and evaluation of a simulator that generates plausible data for the debugging and testing of situation-aware monitoring systems. The use of simulated data is necessary because there is very little sensor-level behavioural-data available in the literature and, moreover, these data are very specific of the set-up used to collect them, i.e. the kind of sensors, the people behavior and age, the environment characteristics, and so on. Moreover, data collection requires a long time since the phenomena we are interested in last months or years. The use of a tool like Repast has substantially sped up the design and implementation phase while still achieving the necessary time performance and quality of the data generated.

Keywords — Agent-based data simulation, Situation Awareness, Situation Understanding, Data Gathering

I. INTRODUCTION

The notions of situation, situation awareness and situation assessment have been formulated by many authors in various contexts.

In his frequently cited paper [1], Endsley presents a theoretical model of situation awareness based on its role in dynamic human decision making in a variety of domains. According to this model, situation awareness is the perception of environmental elements within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future.

The first level of the model consists on perceiving (or gathering) data about the environment. Before the deployment of situation aware systems in real settings, in order to deeply test them, we have to address the difficult problem of gathering or simulating large amounts of data (e.g. years).

In order to do that, we designed and realized an agent-based data-generation software using the Repast Simphony modeling toolkit [2].

This tool efficiently generates large data sets simulating several years of the household activities of an agent-modeled person exploiting the Repast S features.

The remainder of the paper is organized as follows. Section II describes the state of the art of agent-based simulation frameworks. All the details about the data generation tool will be described in Section III. Finally, Section IV details the preliminary evaluation of the tool and draws the conclusions.

II. AGENT-BASED SIMULATION FRAMEWORKS

The first step towards the realization of a data-generation tool for situation-aware systems is the choice of the right simulation framework. We based this choice on the following criteria:

• Modeling power;
• Easiness in respect to new model development;
• Easiness in respect of work need for customizing the tool;
• Programming language used by the developer of the tool.

A taxonomy of selected tools for Agent-based Models (ABMs) can be found in [3]. Following this classification and after having evaluated the environments described in the remainder of this section, we chose to design and implement our data generation tool using the Repast Simphony framework.

Figure 1 shows the original taxonomy extended with the results of our evaluation.

We evaluated the following tools:

Swarm. Swarm [4] is a platform for agent-based models (ABMs) that includes:

• A conceptual framework for designing, describing, and conducting experiments on ABMs;
• Software implementing that framework and providing many handy tools.

As shown in Figure 2, Swarm is based on a nested hierarchy of models with the integration of swarm schedules.

This basic architecture is the simulation of collections of concurrently interacting agents: with this architecture, we can implement a large variety of agent based models. Swarm is the first platform in which this hierarchical organization was introduced. Moreover, it is the first
social complexity simulator, and many other ABMs (e.g., RepastJ, Ascape and MASON) take inspiration from it. Swarm software comprises a set of code libraries which enable simulations of agent based models to be written in the Objective-C or Java computer languages.

*Repast Simphony.* The Recursive Porous Agent Simulation Toolkit (*Repast Simphony* or *Repast S* for short) is a free and open source agent-based modeling toolkit. Similarly to other ABMs, Repast S provides a way to define the so-called “proto-agents”. Generally, the proto-agents are entities with a set of properties and behaviors but without a precise learning behavior. If proto-agents gain learning behavior, they become agents. Conversely, the Repast S framework introduces new modeling structures: *Contexts* and *Projections*. Contexts are the core data structure of Repast S. A Context is a simple container based on set semantics that represents an abstract population of proto-agents without providing any mechanism for their interaction.

Projections are data structures designed to define and implement relationships between proto-agents within a given Context. Projections take the meta-population as defined in a Context and impose a relationship on it allowing interaction between proto-agents. Each Context can have an arbitrary number of Projections associated with it. The Figure 3 shows the interaction between Contexts and Projections. On the left you can see a Context filled with agents whereas on the right examples of Projections (a grid, a network and a map).

*Ascape.* Ascape [5] is a tool for developing and exploring general-purpose agent-based models. In Ascape, a high-level framework supports complex model design, while end-user tools make it possible for non-programmers to explore many aspects of model dynamics. Ascape is written entirely in Java.

*MASON.* Multi-Agent Simulator Of Neighborhoods (*MASON*) [6] is a discrete-event multi-agent simulation library core in Java, designed to be the foundation for large custom-purpose Java simulations, and also to provide more than enough functionality for many lightweight simulation needs. MASON contains both a model library and an optional suite of visualization tools in 2D and 3D.

*NetLogo.* NetLogo [7] is a programmable modeling environment for simulating natural and social phenomena. NetLogo is particularly well suited for modeling complex systems developing over time. Modelers can give instructions to hundreds or thousands of agents all operating independently. This makes it possible to explore the connection between the micro-level behavior of individuals and the macro-level patterns that emerge from the interaction of many individuals. NetLogo is based on a previous modeling environment called *StarLogo*.

In this section, we do not pretend to compile an exhaustive list of simulation, data generation and verification frameworks that are not based on agent models and that can be used for intelligent-environment data generation.

However, in our evaluation we considered a few significant examples. In the paper [8] the author gives a good introduction to data generation and verification tools. The paper describes and we evaluated:

*FADG.* The Flexible Artificial Data Generator (*FADG*) is a data-generator program. Its generic component deals with simulation aspects (event management, events, time stamping, etc.) whereas the domain specific component provides the knowledge of a particular environment (layout, devices, etc.). It can generate events at any level of frequency with a good efficiency (several years of data in a few minutes).

*Spin.* Spin [9][10] is a popular open-source software tool that can be used for the formal verification of distributed software systems. Modeling intelligent environments is possible using the PROcess MEta-LAnguage (Promela) provided by Spin.

*UPPAAL.* UPPAAL [11][12] is a tool for automatic verification of safety and bounded liveness properties of networks of timed automata implemented in C. The tool could be used to model, simulate and verify intelligent environment systems.

Other interesting examples of data-generation frameworks can be found in [13], [14] and in [15].

**III. Data Generation with Repast Simphony**

This paper describes a tool that efficiently generates large data sets simulating several years of the household
activities of an agent-modeled person.

The main agents involved in the simulation are:
- **Person.** This agent models the behaviour (movements, perceptions, etc.) of a person living in the simulation environment.
- **FloorCell, EnvironmentCell, LightCell and HeatCell.** These agents are used to model the behaviour of specific aspects of the simulation environment.
- **LocalizationSensor, EnvironmentalSensor.** These are the sensors deployed in the simulation environment.
- **DayManager, HeatManager.** These agents are used to model the daylight and the heat evolution during the data-generation.

### A. Definition of the Simulation Scenario

Providing a way to define a simulation scenario is the first step towards the creation of a data-generation tool. In order to fulfill this goal, we specified an XML-based scenario-definition file. This scenario file is loaded and parsed during the initialization phase of the Repast S simulator. As described by the following excerpts of the XML definition, it is used to define several aspects of the simulation context. For a more formal specification based on the Document Type Definition (DTD) format, see [16].

Listing 1 shows the definition of the wall cells. If specified here, a FloorCell becomes a wall.

```
<scenario> ...
  <wall> ...
    <wall_cell posX="x0" posY="y0"/>
    <wall_cell posX="yN" posY="yM"/>
  </wall>
  <rooms> ...
    <room key="kitchen" name="Kitchen"> ...
      <area key="kitchen" name="Kitchen-Area"> ...
        <room_cell posX="x0" posY="y0"/>
        <room_cell posX="xN" posY="yM"/>
      </area>
    </room>
  </rooms> ...
  <passages> ...
    <passage fromR="outside" fromA="lift" toR="outside" toA="landing"> ...
    <passage_cell posX="x0" posY="y0"/>
    <passage_cell posX="xN" posY="yM"/>
  </passages> ...
</scenario>
```


```
<scenario> ...
  <loc_sensors> ...
    <loc_sensor key="8D:00:00:00:00:01"> ...
      <pir> ...
        <pir_cell posX="x0" posY="y0" orient="E"/>
        <pir_cell posX="xN" posY="yM"/>
      </pir>
    </pir>
    <range> ...
      <range_cell posX="x0" posY="y0"/>
      <range_cell posX="xN" posY="yM"/>
    </range>
  </loc_sensors> ...
</scenario>
```

Listing 3: XML-based Scenario Definition: Environmental Sensors.

```
<scenario> ...
  <env_sensors> ...
    <env_sensor key="8D:00:00:00:00:01"> ...
      <position posX="x0" posY="y0" orient="E"/>
      <sense_cell posX="x0" posY="y0"/>
      <sense_cell posX="xN" posY="yM" orient="W"/>
    </env_sensor> ...
  </env_sensors> ...
</scenario>
```


More interestingly, Listing 1 shows the division in Rooms and Areas of the simulation environment. The *Room* entity identifies rooms in the environment whereas the *Area* entity identifies disjoint areas of interest in a room. Note that the spatial inclusion of areas $A_1, \ldots, A_n$ in a room $R$ is such that $\bigcup_{i=1}^{n} A_i \subseteq R$.

The hierarchical definition of room and areas is intentionally similar to the hierarchical organization used by the DiGS data-gathering system (see [16]) and it allows to define the network clusters for data generation. In order to generate data sets consistent with those of DiGS, the data-generation tool replicates, for many aspects, the behavior of the DiGS data-gathering algorithms (in particular regarding the person’s movements and RSSI computation).

Moreover, Listing 1 shows the definition of particular *Passage* cells that are used to connect different rooms and areas. Partitioning the environment in this way is particularly useful for the situation assessment techniques in order to efficiently compute user’s room changes and walking direction.

Listing 2 shows the definition of the *Localization Sensors*. In addition to the sensor position on the grid and its orientation, the definition specifies the behavior of the passive-infrared movement sensors in terms of cells covered.

Listing 3 shows the definition of the *Environmental Sensors*.
Sensor. During the data-generation phase, these sensors can read the temperature, humidity and brightness values of the EnvCell agent that is positioned as defined by the `<sense_cell>` element.

In conclusion, Listing 4 shows the definition of Light and Heat Sources. These elements define the properties of the objects that are used to manage and propagate the state of light and heat throughout the simulation scenario. Moreover, Listing 4 shows the elements used to define the position and the initial status of Windows and Light Switches in the simulation environment.

B. Context and Projections

The simulation scenario is modeled by a Repast S model that describes a main Context associated with four Projections. Figure 4 shows the main Projections of the model. Three $N \times M$ grid projections are used to define:

- **Floor Plan.** Floors and walls are defined by a grid of FloorCell agents by appropriately setting their boolean property `wall`.
- **Localization Sensor Grid.** The second grid projection is used to position the LocalizationSensor agents in the scenario and furthermore during the localization data generation.
- **Environmental Data Grid.** The last grid projection is used to position the EnvironmentalSensor, LightCell and HeatCell agents in the environment. Moreover the agents use the grid to simulate the propagation of environmental values such as humidity, temperature and brightness in the context.

The **Movement Network Projection** defines two undirected weighted graphs having FloorCell agents as nodes. The first graph is mainly used for the movement of the Person agent computed using Dijkstra’s shortest path algorithm. Therefore, its weights are computed in order to avoid movements across wall cells.

The second graph is mainly used during the RSSI-data generation to verify if a direct line-of-sight between the person and the localization sensors exists. This information is useful in order to better simulate the RSSI propagation model. Therefore, the weights of the second graph are computed without considering the walls. The line-of-sight information is computed comparing the results of shortest path algorithm between a localization sensor and the person obtained using the two graphs. These results will differ only if there is a wall between the localization sensor and the person.

During the initialization phase, the context and the projections of the simulation model are dynamically populated reading the XML-based scenario file using a context-creation class, specifically defined in the Repast S framework. Furthermore, this is not the only method that can be used to configure the simulation environment: Repast S can load the required initialization data in several ways (e.g. using agent definitions directly stored in a relational database or in text files).

C. Event Scheduling

The data-generation is driven by two different event schedulers that are used to: (i) simulate the behaviour and the habits of the person, (ii) simulate light, temperature and humidity evolution during the day.

In order to simulate the behaviour and the habits of a person living and moving in a realistic home environment, we defined a XML-based file that contains a detailed list of both high- and low-level activities that the Person agent is required to execute during the data generation.

As also shown in Listing 5, the first part of the XML definition describes the high level activities. In addition to a starting and an ending time, every Activity is composed of one or more ActionSets. The same ActionSets are composed of several low-level Actions that indeed specifies what the Person agent should do.

The current version of the simulation tool requires a full description of the low-level actions. This is a simple and efficient approach, nonetheless it is not the right solution when dealing with complex day schedules. Enhancing the capabilities of the Person agent in order to independently compute the set of actions needed to fulfill a high-level Activity is the right way to tackle this problem. Despite that, this improvement of is out of the scope of this paper.

The second event scheduler used by the data generation tool is the one controlling the evolution of the natural light during the simulated day. As shown in Listing 6 it is also based on a XML definition file.

D. Agents and other Objects

As previously described, several agents are involved in the data generation. This subsection details the main agents and other notable objects involved in the simulation.

The Person agent follows the previously described activity schedule moving in the FloorPlan grid projection and interacting with other agents (e.g. for opening doors and windows or for turning on and off switches). Therefore, it has the perception of the other objects that exists in
Listing 5: XML-based Person Schedule Definition: Activities and Actions.

Listing 6: An Short Example of Schedule Definition for Daylight, Temperature and Humidity.

whereas PIR and Movement considering if the person is in one of the cells monitored by the sensor, as defined in the XML scenario file. EnvironmentalSensor agents generate temperature, humidity and light measurements periodically examining the state of the associated EnvironmentCell agent (as also described in the XML scenario definition). Furthermore, all the measurements are stored in a relational database.

The DayManager and HeatManager agents are used to manage the evolution of daylight and external temperature and humidity during the data-generation. Using the previously described schedule they modify the state of LightSources and HeatSources objects. The same LightSources and HeatSources modify the state of the associated LightCells and HeatCells. LightSources and HeatSources are associated with Door and Window objects and with this double passage makes it possible to manage the propagation of light and heat according to the state of doors and windows (e.g. the internal light of a room is zero (dark) if its window is closed, even if the external light has high values).

The FloorCell, EnvironmentCell, LightCell and HeatCell agents are arranged in grid projections in order to replicate the behaviour of walls and temperature, humidity, light variations in the simulation environment. This value propagation is implemented by a controlled flooding algorithm [18] [19]. Every agent propagates temperature, humidity and light changes only if it has not already received the same update request. Therefore, these update messages have, in addition to the new light or heat values, an indication of the simulation tick in which they occur.

Figure 5 shows the Repast S toolkit in execution with different projection displays.

IV. Evaluation and Conclusions

In order to evaluate the simulator performance, we have run it on two different computers having the following configurations. The first one (Config. A) is a Mac with a 2.4 GHz Intel Core i5 processor and 4 GB of 1,067 MHz DDR3 RAM. The second configuration (Config. B) is a PC.
TABLE I: Evaluation of Execution Time Required for the Generation of Simulated Scenarios

<table>
<thead>
<tr>
<th>Num.</th>
<th>Duration</th>
<th>Mean Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,000</td>
<td>1 day</td>
<td>143.7 s</td>
</tr>
<tr>
<td>100</td>
<td>1 week</td>
<td>1,253.8 s</td>
</tr>
<tr>
<td>10</td>
<td>1 month</td>
<td>4,984.5 s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>131.1 s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1,144.3 s</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4,081.9 s</td>
</tr>
</tbody>
</table>

with a 2.6 GHz Intel Core 2 Duo processor and 2 GB of 667 MHz DDR2 RAM.

We were able to generate a number of scenarios lasting from one day to one month. In particular, we generated the data regarding a thousand of slightly different scenarios that last one day; one hundred that last one week and ten with one month duration. Table I summarize the mean execution time required for the generation of the data.

Even if in a few situations we found that the agent-modeled person behaves in a partially incoherent way, the generated data are substantially plausible. Moreover, they were useful to test particular and critical situations, e.g. user falls, otherwise impossible to forecast and to extensively verify.

Data generation to validate monitoring and situation recognition systems is very hard but is unavoidable because the evaluation with real data requires months or years. Of course evaluation in the field is always necessary because the simulator data is just too simplistic to stimulate the monitoring system as the real data would. Nevertheless, debugging and pre-evaluation would not be possible without a reasonable amount of simulated data.

REFERENCES


