Design Decisions in the Development of an Agent-Based Simulation for Large-Scale Emergency Response

Glenn I. Hawe, Graham Coates, Duncan T. Wilson and Roger S. Crouch
School of Engineering and Computing Sciences,
Durham University, Durham, U.K.
{g.i.hawe, graham.coates, d.t.wilson, r.s.crouch}@durham.ac.uk

ABSTRACT

As part of ongoing research into optimizing the response to large-scale emergencies, an agent-based simulation (ABS) is being developed to evaluate different rescue plans *in silico*. During the development of this software, decisions regarding its design have been required in order to best satisfy the following specific application requirements: (1) the construction of a sufficiently detailed virtual environment, representing a real geographical area; (2) the programming of a wide variety of agent behaviors using a minimal amount of code; (3) the computational handling of the "large-scale" nature of the emergency; and (4) the presentation of a highly visual user interface, to encourage and facilitate use of the software by practitioners involved in the project. This paper discusses the decisions made in each of these areas, including the novel use of policy-based class design to efficiently program agents. Future developments planned for the software are also outlined.

Keywords

Agent-based simulation, emergency response.

INTRODUCTION

The REScUE research project (Coates et al., 2011), being carried out at Durham University, aims to identify how to respond optimally to rapidly evolving large-scale emergencies, focusing primarily on events within Great Britain (GB). Central to the research is an agent-based simulation (ABS), the purpose of which is twofold:

(1) to simulate the evolution of a large-scale emergency in a virtual environment, and provide information regarding this emergency to a decision support program (also being developed as part of the REScUE project);

(2) to communicate response plans from said decision support program to agents which represent emergency services personnel. Agents then exercise their autonomy in deciding whether to adhere to these plans.

Thus the ABS provides a way to simulate emergencies and evaluate responses *in silico*. This paper is concerned with design decisions made during the development of this ABS. In the remainder of this section, we detail desirable features sought from the ABS; in the next section, the decisions made to achieve these are discussed.

Desiderata

(1) *Environment*: The use of GIS data to construct the virtual environment in an ABS is now commonplace. For example, PLAN-C (Mysore et al., 2006), WIPER (Schoenharl, 2007) and the AROUND project (Chu et al., 2009) each use shape-files to construct virtual environments for rescue scenarios. Using the Robocup Rescue Simulation platform (Skinner and Ramchurn, 2010), Sato and Takahashi (2011) demonstrated that the fidelity of GIS data can have a noticeable effect on simulation results. Therefore the first desideratum for the ABS is the ability to read in a suitably accurate GIS data source, in order to construct a virtual environment in which all the pertinent features of the real environment are represented.

Reviewing Statement: This short paper has been fully double-blind peer reviewed for clarity, relevance and significance.

(2) *Agents*: Each individual in a human population has their own unique set of attributes and behaviours. Not only should the ABS be able to simulate such heterogeneous populations of agents accurately (in terms of both attributes and behaviors), but preferably it should do so using a minimal amount of code.

(3) *Scale*: No universally accepted definition of "large-scale" exists when applied to emergencies. Intuitively, the "large" can refer to either the geographical extent of the emergency (in the ABS, the size of the virtual environment), or to the number of human individuals adversely affected (in the ABS, the number of agents). Expectations of what is meant by "large-scale" are gleaned from two sources: (i) Gad-el-Hak's classification of disaster severity (Gad-el-Hak, 2009) labels an emergency as "large" if it either covers a geographical extent greater than 10 km^2, or adversely affects more than 100 human individuals (further distinctions are made between "huge" and "gargantuan", but these are the minimum criteria for "large") (ii) The U.K. Concept of Operations (U.K. Cabinet Office, 2010) categorizes emergencies by geographical extent and "impact", with larger emergencies (classed as "significant", "serious" or "catastrophic") generally being "cross-region", unless the impact is particularly high, in which case the geographical extent can be single-scene. Whichever definition of "large-scale" is adopted, it is clear that, ideally, the ABS should be able to cope with simulating many hundreds, if not thousands, of agents situated within a virtual environment representing a real geographical area potentially tens of square kilometers in size.

(4) *User Interface*: One aim of the research is that it be of practical benefit. Practitioners from the local Emergency Planning Units (EPUs) and government office are involved in the project. To encourage and facilitate their use of the ABS software, it is desirable that it be easy to use and interpret: e.g. a new emergency scenario should be able to be set up with minimal effort, via intuitive dialog boxes, tailor-made for the application of emergency response. Instructive visualization of the running simulation is also of importance.

IMPLEMENTATION

As no agent-based toolkit offered all the functionality sought, and as a high level of control is required to deliver software tailored to practitioner usage, the ABS is being written from scratch using C++. Via third-party libraries, C++ offers support for all of the design decisions made in this section. In December 2010, Repast SC++ was released (Collier and North, 2011). It too is written in C++, and makes use of some of the same libraries, namely MPI (Gropp, 2000) and Boost (Boost, 2011). During future development of our ABS, it is planned to make use of Repast SC++ for comparison purposes.

Environment

The desideratum to use high fidelity GIS data, and the fact that the REScUE project is constrained to modeling emergencies in GB, led us to select the Ordnance Survey MasterMap (OSMM) (Ordnance Survey, 2011) as the data source for constructing the virtual environment. Its topography layer supplies data for over 425 million objects in GB in GML format, at a precision better than 10 cm. In the ABS, a *QGraphicsScene* object from the Qt library (Qt, 2011) is used to store the C++ objects which represent the entities in the environment, as it uses a binary space partitioning algorithm to index their locations; thus the entities in each agent's field of vision may be determined very efficiently. Even with millions of entities in the environment, the *QGraphicsScene::items()* functions in the Qt API can determine the items within a region "within a few milliseconds" (Qt, 2011).

The integrated transport network (ITN) layer of OSMM provides details about the entire road network of GB in the form of "links" and "nodes". These are read in by the ABS and used to create a network using the Boost Graph library (Siek et al., 2002). Agents in vehicles are constrained to move along the edges of this network.

Agents

At an abstract level, the functioning of an agent may be broken down into the sequential execution of three methods: *see*, which determines an agent's percepts; *next*, which uses these percepts to update the agent's state; and *action*,which selects an action for the agent to perform based on the agent's updated state (Wooldridge, 2009). Varying the implementation of these methods yields different implementations of agent. If there are n_1 implementations of *see*, n_2 implementations of *next* and n_3 implementations of *action*, then there are $n_1 n_2 n_3$ different possible implementations of an agent. The Strategy design pattern (Gamma et al., 1994) is one possible way to program these possible combinations. This pattern was used in WIPER (Schoenharl 2007) to implement different types of movement and cell-phone behaviors. We have opted to make use of policy-based class design (Alexandrescu, 2001), the compile-time equivalent of the Strategy design pattern. Combining generic programming and multiple inheritance, it reduces the creation of a particular type of agent to the

specification of a set of template parameters, as shown in the declaration in Figure 1. Each instantiation of the *Agent* class inherits its exact implementation details from its template parameters, which in this context are known as "policies".

```
template<typename SeePolicy, typename NextPolicy, typename ActionPolicy>
class Agent : public AgentObject, public SeePolicy, public NextPolicy, public ActionPolicy
```

Figure 1. Policy-based class design of agents

Currently, three different policies for *see* have been implemented: *rectangularSeePolicy* allows an agent to see all entities in a rectangular region aligned around its current direction of motion; *circularSeePolicy* allows an agent to see all entities in a circular region, centered around its current position; *angularSeePolicy* allows an agent to see all entities in a segment of a circle, centered around its current position and aligned around its direction of motion. Three agents (represented by (blue) dots) which make use of these policies, and their fields of vision, are shown in Figure 2. A (red) line emanating from each agent shows its direction of motion. At each time-step, the entities in an agent's field of vision are determined using the efficient *QGraphicsScene ::items()* methods in the Qt API, as mentioned in the previous subsection.

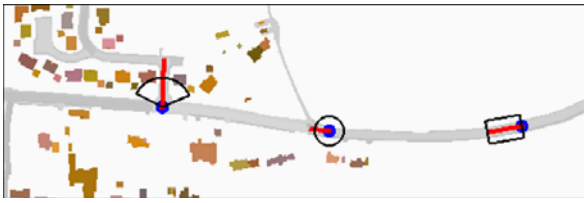


Figure 2. Agents with different "See" policies

Work is currently under way towards implementing different *next* and *action* policies. The use of policy-based class design means that as new policies are implemented, new agent implementations will automatically be possible; furthermore, these implementations will be generated at compile-time rather than run-time. We believe this is the one of the first applications of policy-based class design to the programming of agents within an ABS: with the exception of the unfinished MetaAgent project (De Halleux, 2003), we could find no other reference to policy-based class design being used in ABS.

The National Occupational Standards (NOS) provide descriptions of the activities for different job roles in the UK. These are being used to identify the actions available for selection by each *action* policy; currently, approximately ten actions each for firefighter, paramedic and police officer agents have so far been identified.

Agent-agent communication is implemented using the signals and slots mechanism from Qt. One technicality in this implementation is that a template class (in this case, *Agent*) cannot use signals and slots directly, and so the workaround proposed by Hilsheimer (2005) was used: the *Agent* template class inherits its signal-slots functionality from a subclass of *QObject*. This is the *AgentObject* base class, visible in Figure 1.

Scale

To facilitate the modeling of large geographical areas, MPI (Gropp, 2000) is being used to provide distributed memory parallelism. Large geographical areas are divided up into smaller sub-regions (currently 1 km^2), which are then simulated in separate processes, similar to the work of Koto and Takeuchi (2003). A Cartesian topology is being used to control communications between different sub-regions.

To execute agents concurrently, it is planned to use either shared memory parallelism using OpenMP, or GPU computing using CUDA (Sander and Kandrot, 2010), as recently attempted by Sethia and Karlapalem (2010).

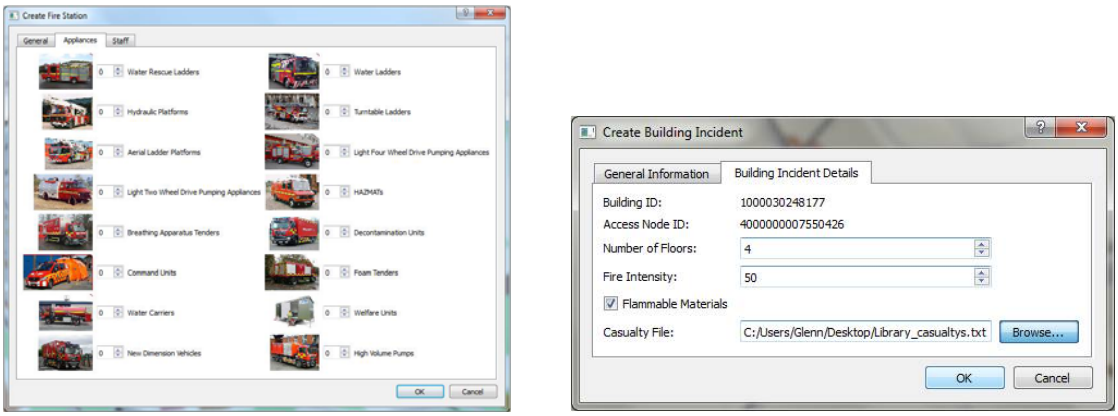
User Interface

A GUI program, shown in Figure 3, has been implemented using Qt. It is a separate program from the ABS (which is a set of MPI processes). The main window of the GUI shows the movement of the agents in their virtual environment. Using sockets, the GUI receives updates on agent positions from each of the individual

ABS processes. Two dockable windows (shown docked on the right hand side of the GUI) allow the user to (1) zoom in to a particular area of interest, whilst maintaining a global view in the main window, and (2) view any area they care to double-click on in Google Street View (Google, 2011) (subject to it being covered).

Figure 3. The GUI program

Context menus in the GUI are used to set up the resources and the crisis. Figure 4 (a) shows the dialog box used to set the attributes of a fire station. Similar dialogs are used to set up police stations, hospitals and ambulance stations. Incidents may be set up at buildings or along roads. Figure 4 (b) shows the dialog box used to set up an incident at a building. It is hoped this easy to use, visual interface encourages eventual usage by the practitioners on the project.



(a) **(b)**

Figure 4. Dialogs for (a) setting the resources at a fire station, and (b) creating an incident at a building

FUTURE WORK

Development of the ABS is ongoing. Future work planned includes: implementation of *next* and *action* policies to create behaviors for both personnel from the emergency services and civilians; use of UK census data to accurately set relevant agent attributes; simulation of static and dynamic emergencies; integration with decision support software (both to supply information regarding the emergency, and receive information regarding response plans). Continued practitioner involvement will help ensure developments remain relevant.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the funding provided by the UK's EPSRC (EP/G057516/1). Further, the authors thank practitioners from the Emergency Planning Units from Cleveland and Tyne & Wear, Co. Durham

& Darlington Civil Contingencies Unit, Government Office for the North East, Fire and Rescue Services from Co. Durham & Darlington and Tyne & Wear, North East Ambulance Service, and Northumbria Police.

REFERENCES

1. Alexandrescu, A. (2001) Modern C++ Design: Generic Programming and Design Patterns Applied. Addison Wesley.
2. Boost (2011) Boost C++ Libraries. Available at: http://www.boost.org/
3. Chu, T., Drogoul, A., Boucher, A. and Zucker, J. (2009) Interactive Learning of Independent Experts Criteria for Rescue Simulations, *Journal of Universal Computer Science*, 15, 13, 2701-2725.
4. Coates, G., Hawe, G. I., Wilson, D. T. and Crouch, R. S. (2011) Adaptive Co-ordinated Emergency Response to Rapidly Evolving Large-Scale Unprecedented Events (REScUE), Proceedings of 8th International Conference on Information Systems for Crisis Response and Management – ISCRAM 2011.
5. Collier, N. T. and North, M. J. (2011) Repast SC++: A Platform for Large-scale Agent-Based Modeling, in Large-Scale Computing Techniques for Complex System Simulations, Dubitzky W., Kurowski, K. and Schott, B. (Eds), Wiley.
6. Gad-el-Hak, M. (2009) The Art and Science of Large-Scale disasters, *Bulletin of the Polish Academy of Sciences: Technical Sciences*, 57, 1, 3-34.
7. Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1994) Design Patterns: Elements of Reusable Object-Oriented Software, Addison Wesley.
8. Google (2011) Google Maps Javascript API v3 http://www.google.com/intl/en_us/help/maps/streetview/
9. Gropp, W. (2000) Using MPI: Portable Parallel Programming with the Message Passing Interface. MIT Press.
10. De Halleux, J. (2003) MetaAgent, A Steering Behavior Template library. Available at: http://www.codeproject.com/KB/library/metaagent.aspx
11. Hilsheimer, V. (2005) Academic Solutions to Academic Problems, *Qt Quarterly*, 15. Available at: http://doc.trolltech.org/qq/qq15-academic.html
12. Koto, T. and Takeuchi, I. (2003) A Distributed Disaster Simulation System that Integrates Sub-simulators. *Proceedings of First International Workshop on Synthetic Simulation and Robotics to Mitigate Earthquake Disaster*.
13. Mysore, V., Narzisi, G. and Mishra, B. (2006) Agent Modeling of a Sarin Attack in Manhattan, *Proceedings of the First International Workshop on Agent Technology for Disaster Management*, Hokkaido, Japan.
14. Ordnance Survey (2011) OS MasterMap – Reliable Spatial Intelligence. Available at: http://www.ordnancesurvey.co.uk/oswebsite/products/osmastermap/
15. Qt (2011) A Cross-Platform Application and UI Framework http://qt.nokia.com/
16. Sander, J. and Kandrot, E. (2010) CUDA by Example: An Introduction to General Purpose GPU Computing, Addison-Wesley Professional.
17. Sato, K. and Takahashi, T. (2011) A Study of Map Data Influence on Disaster and Rescue Simulations's Results. In *Advances in Practical Multi-Agent Systems*, Bai, Q. and Fukuta, N. (Eds). Studies in Computational Intelligence, vol. 325. Springer Berlin/Heidelberg 389-402.
18. Schoenharl, T. (2007) Creating, Updating and Validating Simulations in a Dynamic Data-Driven Application System. PhD thesis, University of Notre-Dame.
19. Sethia, P. and Karlapalem, K. (2010) RoboCup Rescue Simulator on Graphics Processing Units. Available at: http://roborescue.sourceforge.net/2010/tdps/infra/gpu.pdf
20. Siek, J. G., Lee, L-Q. and Lumsdaine, A. (2002) The Boost Graph Library, Addison Wesley.
21. Skinner, C. and Ramchurn S. (2010) The RoboCup Rescue Simulation Platform, *Proceedings of the 9th International Conference on Autonomous Agents and Multi-Agent Systems*, 1647-1648.
22. U.K. Cabinet Office (2010) Responding to Emergencies, the U.K. Central Government Response: Concept of Operations.
23. Wooldridge, M. (2009) An Introduction to Multi-Agent Systems, 2nd Ed., John Wiley & Sons.