THE REPAST SIMPHONY RUNTIME SYSTEM

M.J. NORTH,* Argonne National Laboratory, Argonne, IL,
and The University of Chicago, Chicago, IL
T.R. HOWE, Argonne National Laboratory, Argonne, IL
N.T. COLLIER, Argonne National Laboratory, Argonne, IL,
and PantaRei Corp., Cambridge, MA
J.R. VOS, Argonne National Laboratory, Argonne, IL,
and the University of Illinois at Urbana-Champaign, Urbana, IL

ABSTRACT

Repast is a widely used free and open-source agent-based modeling and simulation toolkit. Three Repast platforms are currently available, namely, Repast for Java (Repast J), Repast for the Microsoft .NET framework (Repast .NET), and Repast for Python Scripting (Repast Py). Each of these platforms has the same core features. However, each platform provides a different environment for these features. Taken together, the Repast platform portfolio gives modelers a choice of model development and execution environments. Repast Simphony (Repast S) extends the Repast portfolio by offering a new approach to simulation development and execution. The Repast S runtime is designed to include advanced features for agent storage, display, and behavioral activation, as well as new facilities for data analysis and presentation. This paper introduces the architecture and core features of the Repast S runtime system and discusses how Repast S fits within the larger Repast portfolio. A related paper in this Agent 2005 conference proceedings by the same authors, "Repast Simphony Development Environment," discusses the Repast S model authoring system.

Keywords: Agent-based modeling and simulation, Repast, toolkits, model execution, runtime system

INTRODUCTION

Repast is a widely used free and open-source agent-based modeling and simulation toolkit (ROAD 2005; North et al. 2006). Three Repast platforms are currently available, namely Repast for Java (Repast J), Repast for the Microsoft .NET framework (Repast .NET), and Repast for Python Scripting (Repast Py). Each of these platforms has the same core features. However, each platform provides a different implementation environment for these features. Taken together, the Repast platform portfolio gives modelers a choice of model development and execution environments.

Repast Simphony (Repast S) extends the Repast portfolio by offering a new approach to simulation development and execution. Repast S runtime is designed to include advanced features for agent storage, display, and behavioral activation, as well as new facilities for data analysis and presentation. This paper introduces (1) the architecture and core features of the

* *Corresponding author address*: Michael J. North, Argonne National Laboratory, 9700 S. Cass Avenue, Argonne, IL 60439; e-mail: north@anl.gov.

Repast S runtime system, and (2) discusses how Repast S fits within the larger Repast portfolio. A related paper in these conference proceedings (North et al. 2005) discusses the Repast S model authoring system.

It is important to note that Repast S and its related tools are still under development. This paper presents the most current information as of the time of its writing. However, changes may occur before the planned final release.

RELATED WORK

There are a variety of existing agent-based modeling toolkits. Repast J, Repast Py, Repast .NET, NetLogo, and Swarm are just a few examples (ROAD 2005; Wilensky 1999; SDG 2005). The growing agent-based modeling literature suggests that the existing toolkits have been useful for many modelers. However, more is needed:

- There is a need to eliminate the restrictions applied by many existing toolkits such as the need to implement interfaces, extend classes, or manage proxies to access specific toolkit functions.

- There are continuing needs to reduce the distance between modelers and programmers, automate common tasks, and encourage the development of flexible, reusable components.

- There is a need to directly support model and enterprise information system integration.

Repast S is explicitly designed to meet these needs.

DESIGN GOALS

The design goals of Repast S directly address the needs identified in the previous section. The design goals for Repast S are as follows:

- All of the core features and capabilities of Repast J and Repast. NET should be available.

- There should be a *strict separation* between models, data storage, and visualization.

- All user model components should be "*plain old Java objects*" (POJOs) that are accessible to and replaceable with external software (e.g., legacy models and enterprise information systems).

- Common tasks should be *automated* when possible.

- *Imperative* "boilerplate" code[1] should be eliminated or replaced with *declarative* runtime configuration settings when possible.

- *Idiomatic* code expressions[2] should be *simple and direct*.

THE REPAST S MODEL IMPLEMENTATION BUSINESS PROCESS

Based on the design goals in the previous section, the Repast S model implementation business process is as follows:

- The modeler creates model pieces, as needed, in the form of POJOs, often using automated tools.

- The modeler uses declarative configuration settings to pass the model pieces and legacy software connections to the Repast S runtime system.

- The modeler uses the Repast S runtime system to declaratively tell Repast S how to instantiate and connect model components.

- Repast S automatically manages the model pieces based on (1) interactive user input and (2) declarative or imperative requests from the components themselves.

The POJO model components can do anything, but are most commonly used to represent the agents in the model. For example, the model components might represent people, organizations, animals, or markets.

The POJOs can be created using any method. The Repast S development environment plans, in particular, feature a set of Eclipse (Eclipse 2005) plugins that simplify both the creation of model components and the specification of model configurations or settings. This functionality is discussed in these conference proceedings in North et al. (2005).

ATOMS AND ATOMIC BONDS

Components are the "atoms" of Repast S models. The components that make up a Repast S model are managed using settings. Settings play an important role in Repast S, since they link the POJO model components to one another and to the surrounding model. Thus, if components are like atoms, then settings are like atomic "bonds." These bonds are used to link the atoms or components to form "materials" or models. Just like real atoms, Repast S model components can be combined together in many different ways to form a variety of models. Settings are designed to bond Repast S atoms together by several means, including:

---

1 Much like boilerplate text, boilerplate code is moderate- to large-sized pieces of code that are used again and again in model after model. An example in Repast J is code to setup a master time schedule or configure a network display.

2 Idiomatic code expressions are short pieces of code that act as slang terms or shorthand. An example in Repast J is code to update the weight on a network edge.

- A model and component query capability,
- Network definitions,
- Spatial neighborhood definitions,
- Enterprise data sources, and
- Time scheduling.

Settings are designed to combine with the Repast S remote access and legacy integration tools to allow external models and enterprise information systems to be treated much like any other software component. There are two categories of Repast S settings, both of which are stored in XML files:

- Model descriptor settings are designed to specify the types of components (e.g., Java classes or legacy models) and the kinds of relationships (e.g., networks) that are allowed in a given model, as well as the declarative triggers or "watchers" to be created (e.g., to let human agents know when there is another human agent near them). Model descriptors define what can be present in a model.
- Scenario descriptor settings are designed to specify such things as the data source for the model (e.g., a set of files), the visualizations for the model (e.g., the two- or three-dimensional agent displays), and the logging to be performed (e.g., results files or charts). Scenario descriptors define what is present in a model.

Model descriptor settings are designed to use reactive planning mechanisms to facilitate the declarative specification of models. The reactive planning mechanisms can include rich event triggers or watchers that use queries to define the components to be watched and implicit declarations to specify the component to be notified. Watchers allow components to declaratively monitor changes in other components.

INTERFACE

The Repast S runtime interface is shown in Figure 1. A very basic model called the "Simple Happy Model" is displayed. The lower side panel titled "Scenario Tree" shows the scenario descriptor, while the buttons and menu on the top left show the simulation controls. The model descriptor (not shown) is used to determine what agent types (e.g., simple happy agents), agent relationships (e.g., a network), and watchers (e.g., update our happiness level whenever a linked friend updates its happiness) are present in the model. Finally, a three-dimensional visualization of a small agent friendship network is shown in the lower right.

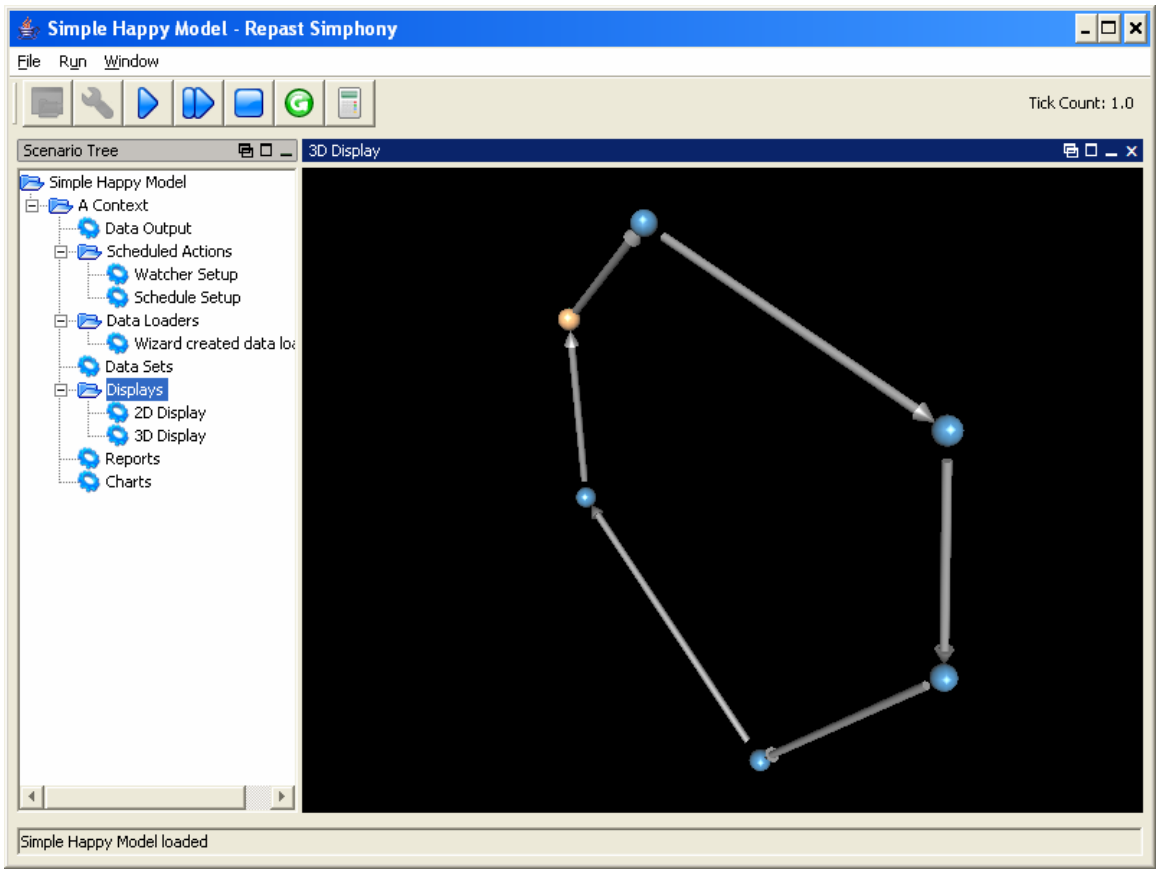

FIGURE 1 The Repast S runtime interface with the Simple Happy Model example

The interface shown in Figure 1 was built using the Simphony Application Framework (SAF). The SAF is a pure Java application development framework intended to simplify the creation of graphical desktop applications. SAF uses Java Foundation Classes, FlexDock (FlexDock 2005), and the Java Plugin Framework (JPF) (JPF 2005) to present and manage application plugins. SAF includes plugin life-cycle management and useful features such as tear-away and dockable windows. Figure 1 shows a set of Repast S windows in the SAF main docking frame. Figure 2 shows a tear-away two-dimensional display. SAF was developed by the Argonne Repast team to support systems such as Repast S. SAF is expected to be released as a separate free and open-source project that supports Repast S.

The Repast S runtime system is designed to include a range of built-in features beyond the previously discussed setting management system. These extended features are organized into Repast S runtime plugins. The planned plugins are expected to work within the SAF-based Repast S runtime interface. The runtime plugins are designed to provide functions such as statistical analysis, interactive charting, and narrative logging. For example, the Repast S plugin

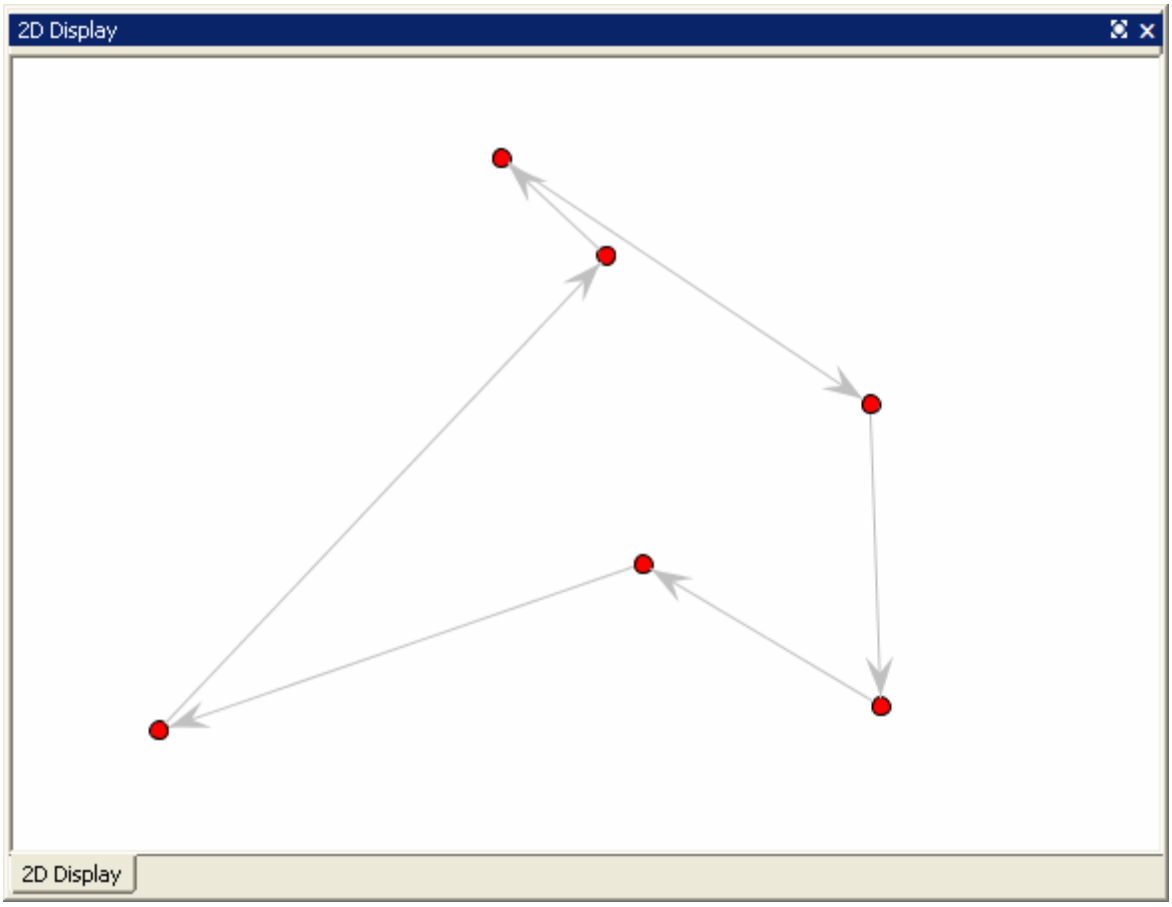

FIGURE 2 A Repast S "tear-away" window with a two-dimensional network visualization

for the R statistics package (R 2005)[3] automatically takes outputs from the Repast S high-performance logging framework and provides them to R Commander[4] (Fox 2005; R Commander 2005) based on interactive user requests. The user provides such requests declaratively in the scenario specification. In this way, Repast S runtime is open to extension, and thus more advanced users are free to add whatever features they may feel necessary.

CONCLUSIONS

The Repast S runtime system is a pure Java extension of the existing Repast portfolio. As discussed above, the Repast S runtime system is designed to provide exciting new features and capabilities to the Repast family. However, Repast S does not replace the existing tools, but

---

3 R and R Commander are both covered by the GNU General Public License (GPL). Software that uses GPL software is itself expected to be covered by the GPL, unless the GPL software is invoked as a clearly identified and separate program. The Repast S plugin architecture can invoke clearly identified and separate programs using a point-and-click interface both for flexibility and to meet licensing requirements such as those for the GPL.

4 R Commander is a point-and-click shell for the R statistical package.

rather is complementary to them. Many users of Repast J, Repast Py, and Agent Analyst may find that porting their models to Repast S is quite straightforward and results in significant simplification and substantial increases in flexibility. Of course, not all existing models should be ported. The users of these models should rest assured that every effort is expected be made to maintain their platforms.

ACKNOWLEDGMENT

The authors wish to thank David L. Sallach for his visionary leadership in founding the Repast project and Charles M. Macal for sustaining involvement in the project. This work is supported by the U.S. Department of Energy, Office of Science, under contract W-31-109-Eng-38.

REFERENCES

Eclipse, 2005, *eclipse*, Eclipse home page, Eclipse Foundation Inc., Ottawa, Ontario, Canada; available at http://www.eclipse.org.

FlexDock, 2005, *flexdock*, FlexDock home page; available at https://flexdock.dev.java.net.

Fox, J., 2005, "The R Commander: A Basic-Statistics Graphical User Interface to R," *Journal of Statistical Software*, Vol. 14, I. 9, Sept.

JPF, 2005, *Java Plug-in Framework (JPF) Project,* JPF home page; available at http://jpf.sourceforge.net.

North, M.J., T.R. Howe, N.T. Collier, and R.J. Vos, 2005, "Repast Simphony Development Environment," in C.M. Macal, M.J. North, and D. Sallach (eds.), *Proceedings of the Agent 2005 Conference on Generative Social Processes, Models, and Mechanisms*, ANL/DIS-06-1, co-sponsored by Argonne National Laboratory and The University of Chicago, Oct. 13–15.

North, M.J., N.T. Collier, and J.R. Vos, 2006, "Experiences Creating Three Implementations of the Repast Agent Modeling Toolkit," *ACM Transactions on Modeling and Computer Simulation* **16**(1):1–25, ACM, New York, NY, Jan.

R, 2005, *The R Project for Statistical Computing*, R Project home page, R Foundation for Statistical Computing; available at http://www.r-project.org.

R Commander, 2005, *R Commander Project*, R Commander Project home page; available at http://socserv.socsci.mcmaster.ca/jfox/Misc/Rcmdr.

ROAD, 2005, *Repast*, Repast home page, Repast Organization for Architecture and Design, Chicago, IL; available at http://repast.sourceforge.net.

SDG, 2005, *Welcome to the Swarm Development Group Wiki!*, Swarm home page, Swarm Development Group, Santa Fe, NM; available at http://www.swarm.org/wiki/Main_Page.

Wilensky, U., 1999, *NetLogo*, Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.