SABLE: Agent Support for the Consolidation of Enterprise-Wide Data-Oriented Simulations

Brian Blake
The MITRE Corporation
Center for Advanced Aviation System Development
1820 Dolley Madison Blvd.
McLean, VA 22102-3481
bblake@mitre.org

ABSTRACT

Simulations are programs that model some real world situation and mirror their interactions in the software domain. In most cases, conclusions can be made based on the analysis of initial and resulting data of the simulation. Consequently, large amounts of data are processed and stored, as both inputs and outputs of the simulation. Handling data is common to most simulation domains, however there is limited specialized support for simulation-based data management. Most simulation environments utilize batch files (large text files) to both supply initial data and record resultant data. This solution is quick and accessible, but there are several problems. One problem is standardization. Even within small corporations there can be multiple data formats for batch files, reducing interoperability between simulation programs. A second problem is the tightly coupled dependency between the correct operation of the simulation and consistency of the file formats and file directory structure. Moreover within one corporation, there can be redundant implementations of file streaming methods. Developers also neglect the benefits of persistence and concurrency that can be provided by a Database Management System (DBMS). This paper introduces SABLE (**S**imulation-oriented **A**gent-**B**ased **L**ibrary and **E**nvironment) as a solution to these problems. SABLE is an agent-controlled environment that enforces the policy of a specialized data format, integrates a DBMS, and removes the task of data management from the developers of the simulation program.

Keywords
Database Management, Agent-Based Systems, CASE Tools

1. INTRODUCTION

Simulation software typically exists in a perennial "prototype" stage. As the analysis needs change, simulations are evolved to allow for changes of both input and output data. Thus simulation environments are dynamic, in nature. In research organizations, functions performed in one simulation may overlap the functions performed in another. Moreover, there are other domain-specific requirements that must be met to ensure consistency in supporting data management across all simulation environments.

Requirements for Data Management in this Domain

1. A standard data format
2. A method for generating new data schemas
3. Support for graphical depiction of data
4. Backward compatibility
   - Ease of integrating with previously developed batch-file based simulations
5. Integration with a DBMS
   - Flexibility to support both object and relational databases.

One requirement that is not solved here is the ability to dynamically take a user directed data schema and produce a database schema. This transformation is dependent on a human in the loop to create the database schema from SABLE's standard data format.

Agent-based software systems have experienced increased attention in the development and support of dynamic systems [1] [2]. Agents have many definitions in industry and academia. One commonality between agent definition is that they act autonomously and have an understanding of their environment. In our case, agents are software components that act as proxies for existing software components or human users. These agents can assume some of the responsibility of system users. In our case, agents are mediators between a user and the data schema needed to run a simulation.

An agent-based solution is applicable for simulation-based data manipulation because the agents can be programmed to encapsulate the data management protocols (i.e. standard data format and data generation methodology). The agent can also enforce the methodology and format. As the system evolves, the agent has knowledge of previous formats and future expectations and is able to manage the data schema efficiently based on knowledge of the data generation protocol.

This paper presents the SABLE framework and architecture in Section 2. In Section 3, there is a depiction of the SABLE process. In Section 4, the SABLE process is illustrated in the context of a particular simulation. Section 5 summarizes this work and its relevance. Finally, future work is presented in Section 6.

2. SABLE ARCHITECTURE

SABLE is an agent-based solution that provides multiple functions to support data management in simulations.

Functions supported by SABLE

- Graphical User Interface (GUI) to input data schema
- Data Schema maintenance as text files
- GUI to input raw simulation data
- Libraries for inclusion in the simulation to encapsulate data input/output
- Functions to integrate new data schema into the library

The SABLE architecture is illustrated in Figure 2.1.

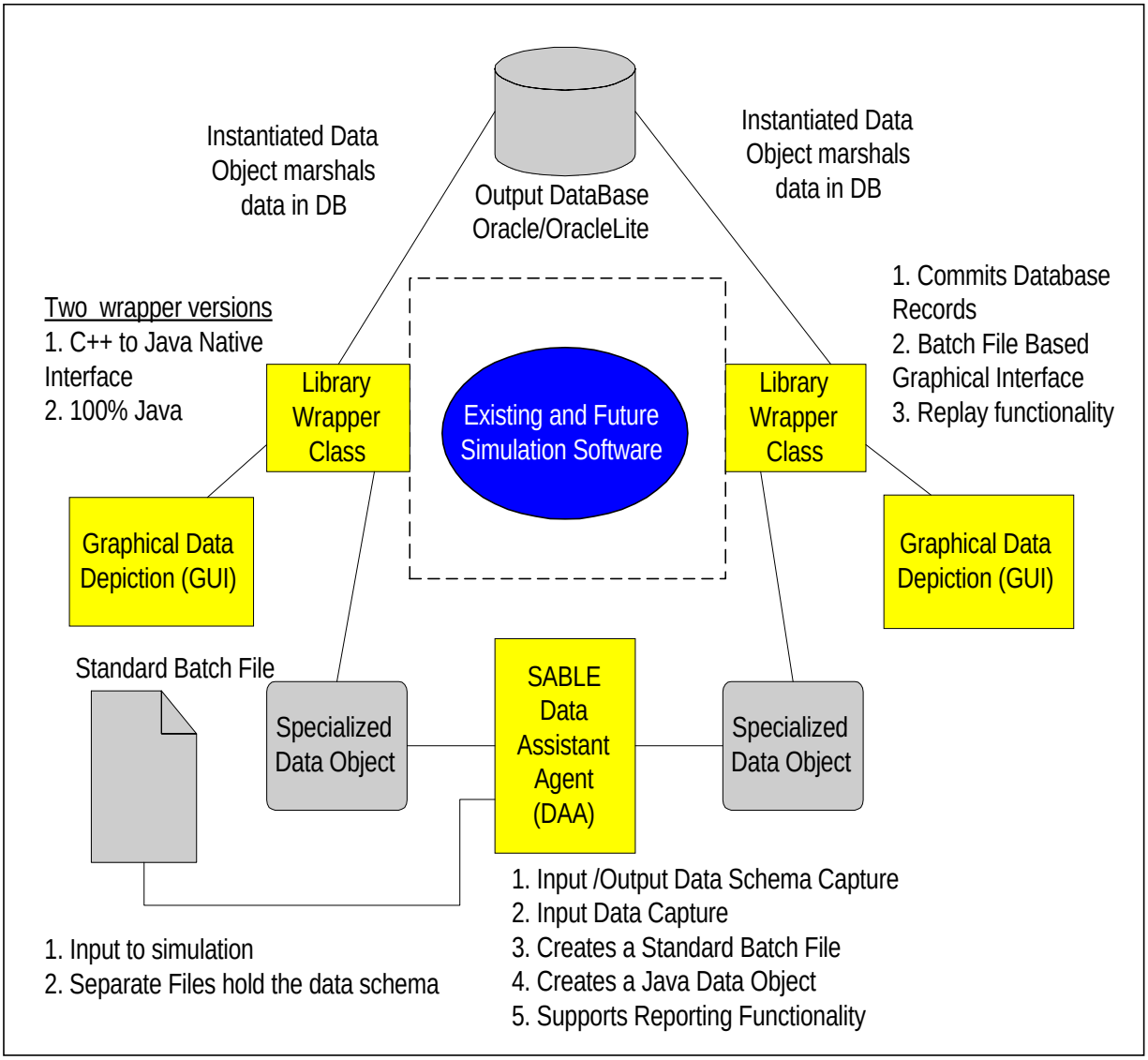

Figure 2.1. SABLE Architectural Components

As seen in Figure 2.1, the SABLE framework consists of the Data Assistant Agent (DAA), a library which contains wrappers for both the input and output of data, and a graphical user interface to show data as it is streamed. Dynamic portions of the framework are the specialized data objects. These data objects are user-specified software objects that realize the data schema in the simulation software via the SABLE library. Finally, there are text (batch) files that store the data schema file-based input data. The central component of the architecture is the Data Assistant Agent. The DAA coordinates all components of the framework.

3. SABLE PROCESS

In understanding the SABLE architecture, one must be introduced to the process. The initial assumption is that the SABLE libraries are installed on the local operating system of the software simulation groups. The first step in the SABLE process is to input the data schema specific to the simulation. The simulation developer executes the DAA to input or update the data schema. The DAA provides a GUI to allow the developer to input the name and type of each data entity. The developer can also aggregate data entities into specialized groups. The DAA will group aggregated data entities into individual software objects. Once the data schema has been captured, the DAA builds two data artifacts, a text file of data schema and a software object of the data schema. Both artifacts are divided into separate files/objects based on specialized grouping. The DAA then includes the specialized data objects in the SABLE library source code and rebuilds the SABLE libraries.

At the time of this writing, the developer is required to create the database schema. The next phase in SABLE's development is to provide an automated method to translate user-prescribed data schemas into a database schema that is consistent with both object and relational database schemas. At this point, the assumption is made that the database model has tables and fields that

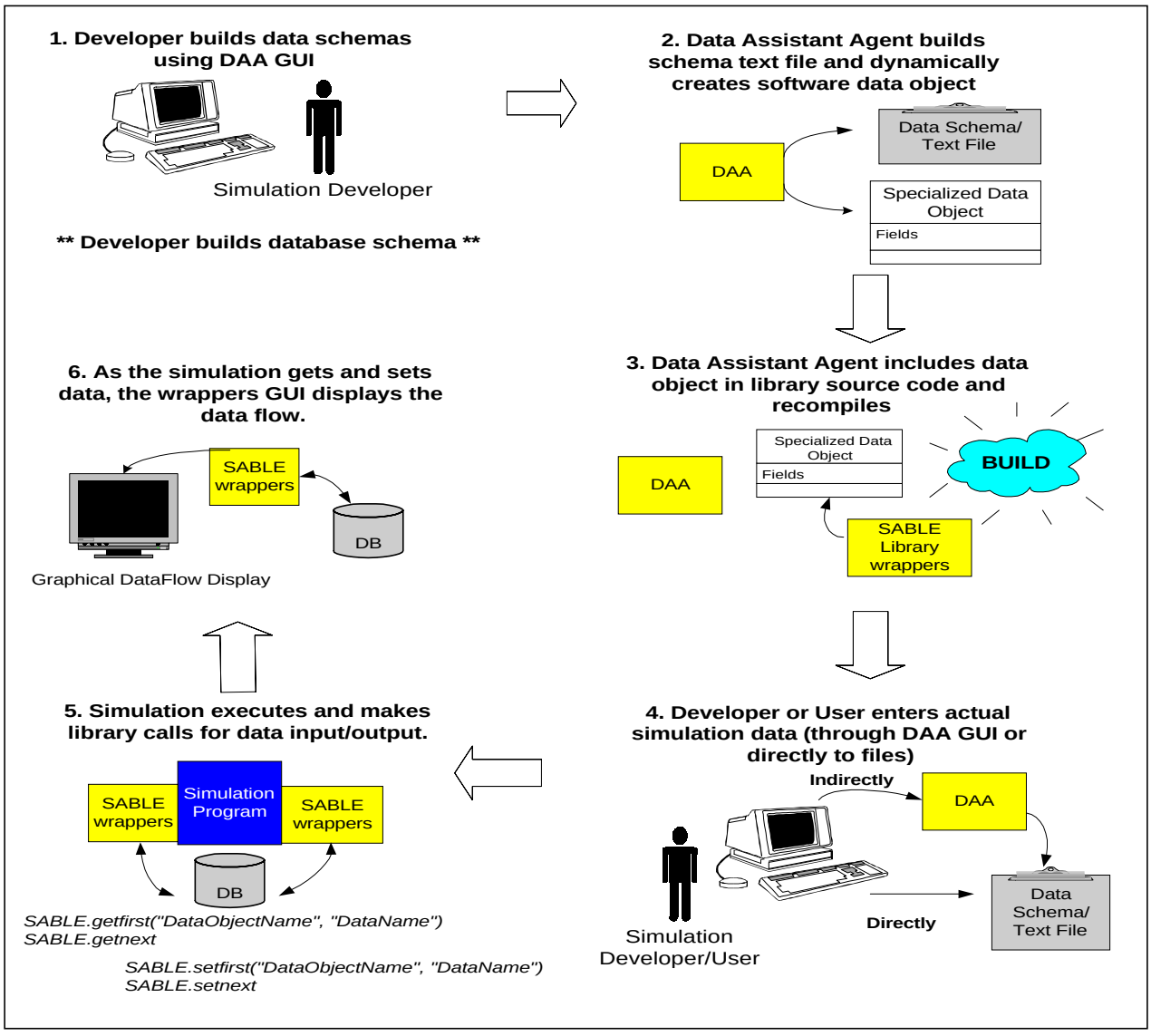

Figure 3.1 SABLE Process

are the same as the object and attributes of the specialized data objects. Database packages currently translate that information into the relational schema needed to do the proper analysis.

At this point, the developer adds the wrapper calls into the simulation software to access the data entities by row. The developer also includes the SABLE libraries. Once the updates have been made, the developer re-compiles the simulation software. The final step is actually entering the raw data for the simulation. The developer/user has two options. One option is to edit the data schema text file by hand. This option is allowed just in case there is a large amount of data that has to be translated from other formats. The DAA allows the user/developer to enter the information via a GUI. The DAA then takes the information and writes it to the data schema text files. At this point, the system is ready to execute. This process is depicted in Figure 3.1

As the system executes, initial data are read in from the data schema text files. The software objects are instantiated by the SABLE wrappers and populated. As the main simulation reads the data from the software objects, there is the option to display the data flow on external graphical interfaces. The GUIs show a basic flow of data but can be manipulated to show the data in more complex formats. The main simulation populates the software objects with output data. At the same time, the software objects store these outputs in the database for later analysis. Subsequent simulation runs can pull data directly from the database (this supports the playback capability). The simulation saves data to the database in the static schema constructed by the developer.

4. SAMPLE CASE FOR IMPACT

The MITRE Corporation has an agent-based simulation, IMPACT, that mirrors the air traffic management problem domain. IMPACT takes initial flight schedules among other operational parameters (i.e. ground delay programs, ground stops, flight change, airline information, etc.) and determines the actual simulated departure/arrival times into a particular airport. Here a subset of IMPACT is used to show a sample case of the SABLE process. In this case, let's assume that that the input data schema of IMPACT is the same as the output data schema; in reality there is some variation. The data schema would incorporate the airline name, departure time, arrival time, and arrival airport (this is a subset of the actual data needed in the simulation). The DAA would collect the information tabulated in Table 4.1.

Object Name	Number of Entities
Schedule	4

Data Entity	Name	Data Type
Entity_1	AirlineName	Char[10]
Entity_2	DepartureTime	Int
Entity_3	ArrivalTime	Int
Entity_4	ArrivalAirport	Char[4]

Figure 4.1 SABLE-DAA data schema

The DAA would then create a text file that contains the data schema. This text file will allow the users of IMPACT to enter information directly or indirectly (i.e. via the DAA). The file will be named as the object name and the first line will be the data schema. Also, once the software data object is instantiated in IMPACT, it will read initial data from this text. This text file format is depicted below. Percent sign (%) delimiters can be replaced for any delimiter.

Header line

Airline:char[10]%DepartureTime:int%ArrivalTime:int %ArrivalAirport:char[4]

Text Line

Entity1_value%Entity2_value%Entity3_value% Entity4_value%

Finally, the DAA dynamically creates a software data object that is to be included in SABLE wrapper libraries. During operation, IMPACT instantiated these software objects. These software objects encapsulate all the functionality of persistence, concurrency, and graphical data display. The main simulation only has to set a couple of parameters and make some generic function calls. Sample software object source code and function calls are depicted below.

Schedule Data Object

```
public class Schedule extends
SABLE_DataObject
       {
        private string AirlineName[]
        private int DepartureTime[]
        private int ArrivalTime[]
         private string ArrivalAirport[]
       }
```

Simulation Code Sample

```
// Instantiate Data object
Schedule this_sched = new schedule();

// Get data values for Airlines
this_sched.getfirst(Schedule,
AirlineName)

// Iterate through data values
while(this_sched.getnext(Schedule,
AirlineName) )
……
```

5. SUMMARY

SABLE is an agent-based solution that provides an approach to the standardization of data management in the dynamic domain of simulation development and evolution. Design decisions for SABLE were based on a great deal of low-level design consideration, which is neglected in this paper. Only a small amount of low-level design could be detailed in the space of this paper. The strength of this approach is the ease

of use offered by SABLE. The use of a DBMS has been abandoned in the context of most air traffic management simulations because of time constraints and the learning curve associated with database connectivity and database schema. The functionality of the SABLE system is similar to functionality provided in many component libraries as in Microsoft Foundation Classes (MFC). The main benefit realized with the SABLE approach is for large-scale simulation development environments. Other component libraries are not specifically suited for integration, but more for "top-down" development.

This first phase of SABLE attempts to create and enforce a universal data management process to be integrated in past, existing, and future systems. SABLE has also been incorporated into 2 other simulations. The integration of SABLE objects decreased development time approximately 35% compared to parallel projects incorporating the same functionality without SABLE objects. The main time constraint in the process was designing the data model that would be used by SABLE. Iterative integration of SABLE objects into future projects will further strengthen and formalize SABLE's data management functionality.

6. FUTURE WORK

The overall goal of the SABLE project is to provide an agent-based toolkit to allow modelers to make use of agent-oriented development techniques in their simulations. This toolkit is being developed in a phased approach. The four phases are data management, agent communication, policy/action engine development, and learning/inference engine development. The work in this paper specifically documents the first phase of data management. The future SABLE agent will be used as autonomous entities in simulations that perform calculations and produce actions, based on beliefs extracted from a common data model. The SABLE data objects will allow these future agents to introspect this data model and infer various beliefs. The forthcoming SABLE agents will have a modular structure as illustrated in Figure 6.1. The SABLE agent will be composed of various components for event-based communication, data/policy management, and a policy execution engine.

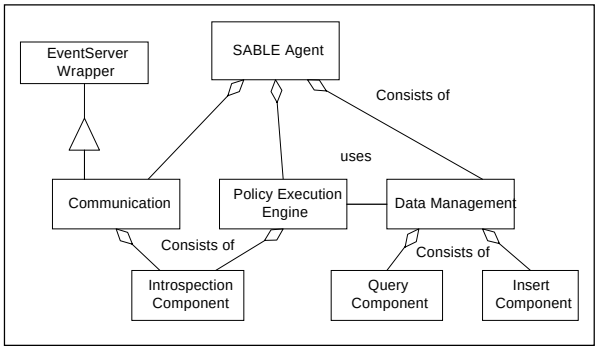

Figure 6.1 Forthcoming SABLE Agent Structure

There are some areas of improvement in this first phase of the SABLE project. The main area is creating a universal database format from which the DAA can transform the developer-prescribed schema to a database schema. Also, additional design is needed to maintain a consistent and efficient graphical view of the input and output data. Finally, for time critical simulations, SABLE will further loosely couple [3] the act of committing data to the database by putting them in a separate process.

7. REFERENCES

[1] Blake, B. and Bose, P., 2000. An Agent-Based Approach to Packaging Mismatch. *In Proceedings of the 4th International Conference of Autonomous Agents (AGENTS2000)*, Barcelona, Spain

[2] Graham, J. and Decker, K., 1999. Towards a Distributed, Environment-Centered Agent Framework. *In Proceedings of the 1999 Intl. Workshop on Agent Theories, Architectures, and Languages [ATAL-99]*, Orlando, Fl

[3] Shaw, M. and Garlan, D., 1996. *Software Architectures: Perspectives on an Emerging Discipline*, Prentice-Hall

[4] Sun Microsystems Inc., The Java Language Specification, Distributed Event Model Specification. http://java.sun.com.