MODELING FORCE DEPLOYMENT FROM ARMY INSTALLATIONS USING THE TRANSPORTATION SYSTEM CAPABILITY (TRANSCAP) MODEL

James F. Burke, Jr., Richard J. Love, Charles M. Macal, Dawn L. Howard, and Jill Jackson
Argonne National Laboratory
Decision and Information Sciences Division
9700 S. Cass Avenue, Bldg. 900
Argonne, IL 60439

KEYWORDS

deployment, transportation, discrete-event simulation, logistics

ABSTRACT

Planning a military deployment is a complex, multi-phased problem. Deployment planners must consider the constraints, options, and available infrastructure at each installation, from the beginning of the transportation system in the United States to the end of the deployment in the host country. The Military Transportation Management Command Transportation Engineering Agency (MTMCTEA), in collaboration with Argonne National Laboratory (Argonne), developed the Transportation System Capability (TRANSCAP) model to simulate deployment of forces from army bases. TRANSCAP uses sophisticated simulation processes to conduct detailed analyses, plan real-world operations, and train army transportation specialists. TRANSCAP's design separates its pre- and post-processing modules (developed in Java) from the simulation module (developed in MODSIM III). The pre-processing module uses a graphical user interface (GUI) to manage database and geographic information system (GIS) linkages and to create, modify, and display simulation scenarios. The post-processing module displays the simulation results in reports and graphs. The simulation module contains the heart of TRANSCAP, the discrete-event simulation (DEVS).

This paper describes TRANSCAP's modeling approach, its use of classes with fields and methods specific to army installations, and its reusable deployment simulation framework of classes, the Transportation Class Hierarchy. This framework has been adopted by the Army Modeling and Simulation Office (AMSO) as a standardized code structure for object-oriented deployment simulations to ensure meaningful data exchanges. The framework has proved flexible enough to be reused in other deployment simulations.

INTRODUCTION

During a military deployment, both personnel and equipment are transported from a fort to a tactical assembly area and then to the final destination — the forward line of troops. Such "fort-to-foxhole" deployments are not practiced often because of the cost: when practiced, they are rarely conducted under realistic circumstances. As a result, the deployment analysts are not fully aware of the constraints and complexities of a deployment.

To address this issue, the MTMCTEA and others are developing simulations to analyze military deployments. Simulations are used to analyze, plan, train for, and execute deployments. New simulation models and technologies are a good investment because simulating deployments first on a computer is more efficient and cost effective than testing them in the real world.

MTMCTEA collaborated with Argonne and others to develop a suite of four force projection models (FPMs). The suite is part of a strategic vision for simulating most of the military deployment process. The Enhanced Logistics Intratheater Support Tool (ELIST) simulates the infrastructure between ports and forts. The Port Simulation (PORTSIM) and the Coastal Integrated Throughput models simulate throughput at austere ports. The TRANSCAP model simulates installation transportation operations, computes time-phased outloading capability, compares computed capability to outloading requirements, and identifies system and infrastructure constraints and installation-specific force clearance profiles. Not only is TRANSCAP a component of the FPM suite, but it can be used for fort infrastructure analysis, force deployment analysis, practice, and training.

TRANSCAP is part of a multi-model architecture that simulates a corridor or an entire theater. The model uses sophisticated simulation processes that allow the user to conduct a detailed analysis of the critical installations in a theater of operations. ELIST simulates transporting cargo across U.S. infrastructure to airports and seaports. ELIST also performs an aggregate installation-process simulation for most of the installations in a theater of operations on which the user is not focusing. A similar model suite, with data exchange between ELIST and TRANSCAP, is also being developed for the theater of operations in the host country.

TRANSCAP was written at the lowest possible level of data and process. It was also written in an object-oriented programming language to foster the greatest amount of modularity and potential for reuse for future MTMCTEA simulation projects.

The following paragraphs describe the three types of low-level data that TRANSCAP uses: transportation and cargo, infrastructure, and resource. Transportation and cargo data are Level 6, which means that details are available for each individual piece of cargo and transport vehicle, rather than an aggregated data description based on tonnage or category. Infrastructure data are the most detailed type of data about quantity and the exact measurements of existing infrastructure. For example, rather than using an aggregate capacity for all marshaling areas or rail yards, the model uses specific data for each. Resource data include information about such existing facility resources as the quantity and location of each truck loading ramp, rail end ramp, inspector, mechanic, or tie-down crew.

There are also three types of processes in deployment simulations. TRANSCAP processes are implemented in an object-oriented programming language. These processes are at a low level because of how transportation and cargo, infrastructure, and resources are modeled. Transportation and cargo are modeled at a low level because there is a different class for each type of cargo, transport, and locomotive. Also, each individual piece of cargo, transport, and locomotive is represented as its own object — aggregation is not used. Processes simulating infrastructure utilization are modeled at a low level because the process times for the movements of transports, cargo, and locomotives are based on the actual lengths of the infrastructure that is traversed, rather than on stochastic (or random) distributions. Resources, like transportation and cargo, are modeled at a low level because there is a different class for each type of resource, and each individual resource is represented as an object. Also, each resource is assigned a specific responsibility and a specific service time, multiple processes are never represented by a general rate.

This paper describes TRANSCAP's modeling approach, its use of classes with fields and methods specific to army installations, and its reusable deployment simulation framework of classes, the Transportation Class Hierarchy. This framework has been adopted by the AMSO as a standardized code structure for object-oriented deployment simulations to ensure meaningful data exchanges.

COMPONENT MODULES

TRANSCAP comprises four categories of component modules: pre-processing, simulation, post-processing, and inter-process communication.

- The pre-processing module includes the Installation, Force, and Scenario Manager and the Scenario Report Manager. These pre-processing modules manage data input, select the installation, specify pieces for deployment, identify the method of deployment (e.g., convoy), and specify deployment characteristics (e.g., the method used to form the convoy groups).

- The simulation module contains the heart of TRANSCAP — the DEVS.

- The two post-processing modules include the Results Display Manager, which manages the reporting and presentation of simulation results, and the Export Manager, which exports results to other simulation models.

- The Inter-process Communication Manager enables interaction between the pre- and post-processing modules and the simulation module.

REUSABLE DEPLOYMENT SIMULATION FRAMEWORK

The reusable deployment simulation framework designed for TRANSCAP has been reused in other MTMCTEA and Argonne deployment simulations (Burke *et al.*, 1999). The EXtensive Hierarchy and Object Representation for Transport Simulations (EXHORT) is a collection of three class hierarchies that together constitute a standard and consistent class attribute representation and behavior that could be used directly by deployment simulations.

The reusable deployment simulation framework was extended for TRANSCAP to develop military fort-specific subclasses containing logic for specific interactions between clients, servers, and areas. An example of how default behavior from the reusable deployment simulation framework is subclassed is a vehicle being loaded onto a railcar. The interaction between clients and servers is implemented as a resource holding a piece of cargo in place for a stochastic process time and then updating its current status field when completed. No additional processing is done to the client or the server during the service period. However, when a vehicle (i.e., a client) interacts with a rail end ramp (i.e., a resource) with the goal of loading that vehicle onto the next available position on a train of railcars, the code that chooses a process time stochastically needs to be overridden. The new code (that overrides the old code) will choose a process time on the basis of the length of the railcars and spanners over which the vehicle must drive. In this way, the generic interaction between a client and a server, which is found in the reusable deployment simulation, is subclassed for a specific situation at a fort.

General Transportation Simulation Class Hierarchy

The class abstractions in EXHORT are specified at a detailed level encapsulating deployment simulation characteristics, rather than at the higher, more generic level that is typically used in object modeling.

EXHORT allows deployment simulations to use the same set of underlying class data and significantly reduce the effort needed to integrate simulations and to analyze the defense transportation system. The first hierarchy is the Transportation Class Hierarchy that covers commercial transportation assets and military cargo. The second hierarchy is the Infrastructure Class Hierarchy, which addresses the locations where activities are performed and the physical infrastructure, such as a motor pool at a fort or a berth at a port. The third hierarchy is the Resource Class Hierarchy, which describes the resources needed to support deployment (e.g., ramps, rough terrain container handlers, cranes, and forklifts).

These class hierarchies encapsulate generic interactions: (1) between clients — things that are acted upon (e.g., cargo and transports), (2) between clients and servers/resources — actors performing a service (e.g., inspectors, mechanics, end ramps, and cranes), and (3) between clients and areas at a facility (e.g., landmarks or areas of activity). The client/server interaction is simply a client being served for some stochastic interval, and upon completion, updating the client's status. Examples of an interaction between a client and an area include movement between areas or tracking the constraining factors (such as the maximum number of clients that can wait in line and then wait around after receiving a service). An example of an interaction between servers and areas is tracking the number of servers available in each area (e.g., 41 inspectors in marshaling area #1 and 21 in marshaling area #2) and determining whether the servers (i.e., inspectors) may travel between areas (e.g., inspectors remain in an assigned marshaling area, but tie-down crews move between rail loading yards).

Fort-Specific Subclasses

Argonne assembled classes specific to TRANSCAP and subclassified them from the generic class hierarchies. Together, these eight subclasses create an installation deployment simulation that involves the following nine processes:

- Initialize simulation
- Generate vehicles — vehicles depart motor pool
- Process vehicles
- Generate truck transport assets
- Process truck transport assets
- Simulate container movement
- Generate trains
- Process trains
- Call forward locomotive process

BENEFITS OF MODULE SEPARATION

TRANSCAP generally consists of three phases: (1) obtaining a scenario (pre-processing), (2) simulating the scenario, and (3) providing the results (post-processing). Because the phases are mostly independent, they have convenient boundaries for separation. Within each phase, there are similar boundaries between smaller phases. Separating TRANSCAP into independent phases achieves a division of labor; each phase can be developed as an individual component focused on its own task. Each component may even be written in a language that is best suited for its task, regardless of the language(s) in which other components are written.

Separating the component modules provides a layer of abstraction between the simulation and the data inputs. As data inputs are modified or added, the Installation, Force, and Scenario Manager must also be changed so that it can create installations and scenarios; the DEVS does not need to be changed.

Separation also allows the developer greater independence. Different engineers can design the modules with a greater degree of freedom and focus on each module's task. This freedom leads to another benefit — reusable components. Some modules perform the same, or similar, task, as required by other simulation models. When engineers design for it, the modules can be reused in multiple models. This reuse benefits the other models; conversely, modules from other models can also be easily reused in TRANSCAP. For example, the Results Display Manager was designed as an independent component, and is now used by several models.

Because some modules execute in separate processes, TRANSCAP may be able to take advantage of a multi-processor machine or multiple machines. For example, multiple DEVS modules could simultaneously execute multiple scenarios. The pre-processing modules could be distributed via the Internet, and the DEVS could become a server, thus allowing Web-based simulations. Having separate processors or machines could also assist the implementation of callback animations, which would display the simulation as it executes a scenario.

MULTIPLE LANGUAGES

TRANSCAP takes advantage of the module separation by writing each module in a language best suited to its task. The DEVS is written in MODSIM III. Of all of the object-oriented programming languages, MODSIM features the most complete framework of vendor-supplied classes for DEVS programming. There is almost no need to write statistical collection or DEVS facilitation source code when using MODSIM III. An object-oriented programming language was also selected for its modularity and reusability, which allow the source code to be constructed using a layered approach — ensuring the creation of scalable, portable components for reuse in TRANSCAP's development and in other deployment models.

The Installation, Force, and Scenario; Scenario Report; Results Display; and Export Results Managers (the pre- and post-processing modules) are written in Java 2. The Inter-process Communication Manager utilizes both Java 2 and C. A trait shared by these languages is that they are available on both of TRANSCAP's target operating systems, Solaris (UNIX) and Windows NT.

CONCLUSION

TRANSCAP's architecture has been designed to take advantage of the independent nature of its major processes. Its pre- and post-processing component modules have been separated from the DEVS, simplifying the design and development, providing data abstraction, selecting an appropriate language to be used for each module, and allowing greater freedom for future growth. The Inter-process Communication Manager maintains the cohesiveness of this team of modules. The potential for reuse is enhanced by having and sharing well-defined modules and by implementing AMSO's reusable deployment simulation framework. This approach has resulted in a richer, more dynamic simulation model.

ACKNOWLEDGMENTS

This work was supported under a military interdepartmental purchase request from the U.S. Department of Defense, MTMCTEA, through the U.S. Department of Energy contract W-31-109-ENG-109.

The authors wish to acknowledge the support of our program managers Melvin Sutton and Maureen Cassada of MTMCTEA.

The authors also thank Mary Fitzpatrick of Argonne's Information and Publishing Division for her editorial assistance.

REFERENCE

Burke, J.F., C.M. Macal, M.R. Nevins, C.N. VanGroningen, D.L. Howard, and J. Jackson. 1999. "Standardization of Transportation Classes for Object-Oriented Deployment Simulations." In *Proceedings of the Simulation Interoperability Standards Organization 1999 Fall Simulation Interoperability Workshop* (Orlando, FL, September 12–17). SISO, Inc., Orlando, FL (3) 1142–1152.

AUTHOR BIOGRAPHIES

James F. Burke, Jr., is a software engineer in the Modeling, Simulation, and Visualization Group at Argonne National Laboratory. He is the project leader and lead designer of the TRANSCAP model, which is a part of the Logistics Modeling and Simulation Program. He is working on a Ph.D. at the Illinois Institute of Technology, studying simulation component reuse and standardization. He received an M.S. in computer science from the Illinois Institute of Technology and a B.S. in Computer Science from Benedictine University. His research interests include reuse and standardization of object-oriented simulations, CORBA, simulation modeling, and object-oriented software design and development. He is a member of IEEE.

Richard J. Love, is a software engineer in the Modeling, Simulation, and Visualization Group at Argonne National Laboratory. He is project leader of the 2D Viewer, the animation model employed by the PORTSIM and TRANSCAP models, and leads development of TRANSCAP's GUI. He earned an M.S. in Computer Science and a B.A. in Physics from North Central College and a B.S. in Electrical Engineering from the University of Illinois. His other research interests include object-oriented design, software agents, and Web-enabling technologies.

Charles M. Macal directs the Modeling, Simulation, and Visualization Group and leads the Logistics Modeling and Simulation Program at Argonne National Laboratory. His research interests include simulation modeling and architectures and agent-based modeling. He received a Ph.D. in operations research from Northwestern, as well as degrees from Purdue University. He is a registered professional engineer in Illinois and a member of INFORMS, the American Association for Artificial Intelligence, the Society for Computer Simulation, and Tau Beta Pi.

Dawn L. Howard is a software engineer in the Decision and Information Sciences Division of Argonne National Laboratory. She is the project leader of the CITM model being developed in the Simulation and Visualization Section. She received her M.S. in computer science from the Illinois Institute of Technology and holds a B.S. in computer science from St. Xavier University. She also holds a B.A. in English from DePaul University. Her research interests include object-oriented development and simulation modeling.

Jill Jackson is a knowledge engineer in the Decision and Information Sciences Division of Argonne National Laboratory. She received her M.S. in environmental management from the Illinois Institute of Technology and a B.A. in communication studies from the University of Iowa. Her research interests include communicating advanced computer concepts and applications to users and decisionmakers.