THE TRANSPORTATION SYSTEM CAPABILITY MODEL (TRANSCAP): A MIXED LANGUAGE DEVELOPMENT APPROACH FOR AN ARMY DEPLOYMENT SIMULATION

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

KEYWORDS

mixed language, inter-process communication, class framework, discrete-event simulation

ABSTRACT

The Transportation System Capability (TRANSCAP) model is a discrete-event simulation model designed to simulate deployment of forces from army bases — the first step of an army deployment. Argonne National Laboratory (Argonne) collaborated with the Military Transportation Management Command Transportation Engineering Agency (MTMCTEA) to develop TRANSCAP. The model, which dynamically simulates the loading and transport of military cargo from an installation, will be used to plan real-world operations and to train army transportation specialists. TRANSCAP was designed with pre- and post-processing modules (developed in Java separately from the discrete-event simulation module, which was developed in MODSIM III). Multiple programming languages were used to meet the needs of this simulation. This paper highlights the function of each module, describes how the modules interact, identifies the benefits of the separation, and describes the programming languages used to develop the modules. The paper also discusses a reusable deployment simulation framework of classes that implements the Army Modeling and Simulation Office's (AMSO's) standard Transportation Class Hierarchy. This framework has proved flexible enough to be reused in other deployment simulations.

INTRODUCTION

During a military deployment, both people 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, plan, train for, and execute deployments. New simulation models and technologies are good investments because simulating deployments 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) Model and the Coastal Integrated Throughput models simulate throughput at first-class, commercial ports and 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 departure profiles. Because it simulates the first step of a military deployment, TRANSCAP's main role will be to present these departure profiles to ELIST for simulation of the next leg, which is movement to the port. 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.

This paper discusses the components and function of TRANSCAP's six modules, the benefits of separating the modules, and the three programming languages used to develop the modules.

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 & 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 discrete-event simulation.

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.

Discrete-Event Simulation

The goal of TRANSCAP's discrete-event simulation is detailed analysis. Because ELIST can perform a high-level, aggregate analysis of the operations at a fort, 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 in future simulation projects.

The following are descriptions of 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, rather than an aggregated data description based on tonnage or category. Infrastructure data are the most detailed type of data about quantity and exact measurements of existing infrastructure. For example, rather than using an aggregate capacity for all marshaling areas or rail yards, specific data for each are used. 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 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 instantiated as its own object — aggregation is not used. Processes simulating infrastructure use are modeled at a low level because the process times of the movements of transports, cargo, and locomotives are based on the actual distances in 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 instantiated 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.

Installation, Force & Scenario Manager

The first pre-processing module, called the Installation, Force & Scenario Manager, encapsulates most of TRANSCAP's graphical user interface (GUI). It manages the installations available for simulation, the forces resident at the installations, and the deployment scenarios. This module allows the user to specify and generate the scenarios needed for the simulation module. For TRANSCAP's purposes, an installation (e.g., Fort Stewart) consists of a resident force; a modeled infrastructure consists of landmarks and routes. A scenario is a description of a deployment, which includes the installation from which the items are deployed, the items to be deployed, and other deployment characteristics. A resident force must exist to create an installation, and an installation must exist to create a scenario.

If a desired installation does not exist in TRANSCAP's database, it can be created in the Installation, Force & Scenario Manager module.

Creation of an installation requires the input of force data and detailed infrastructure data or the creation of notional infrastructure data. When creating notional infrastructure, the user must enter the location of the landmarks and the distances of the routes. By using varying force or infrastructure data, the user can create multiple versions of an installation.

The user creates a scenario by selecting an installation, selecting a portion of the installation's resident force to be deployed, selecting the pieces to be deployed, specifying how those pieces will be deployed, and selecting other deployment options (e.g., hours for daylight operations). Most deployment options have defaults, so the user may only need to change a few options. An existing scenario can be displayed for review, modified, and saved as a new scenario, or selected for immediate execution in the discrete-event simulation module. The user can also perform what-if analyses by creating scenarios, varying deployment characteristics, or basing those scenarios on multiple versions of installations.

Scenario Report Manager

Because a scenario is defined by such a large quantity of information, it is useful to view and analyze it in report form. The Scenario Report Manager displays and organizes the existing scenario's raw data into sections and tables. This report can be viewed in a scrollable window on the user's monitor or printed.

Results Display Manager

After the simulation executes a scenario, the results need to be provided to the user. The Results Display Manager presents these results in a variety of reports, graphs, and tables. Each report, graph, or table result is displayed in a separate window and can be printed. The user also has the option to zoom in to highlight specific results.

Export Manager

Other models may need the results of an executed scenario. The Export Manager creates an export file and writes it to a directory location specified by the user. Currently, TRANSCAP only exports files to PORTSIM.

Inter-process Communication Manager

Because a different programming language is used for the simulation module than for the pre- and post-processing modules, and because some of these modules execute in separate processes, a communication mechanism is required to ensure interaction: the Inter-process Communication Manager serves this purpose. The Installation, Force & Scenario Manager tells the discrete-event simulation what scenario file to execute. The discrete-event simulation provides execution feedback to the Installation, Force & Scenario Manager, which displays a percent completion progress bar in a dialog box and provides results files for the Results Display and Export Manager.

The Inter-process Communication Manager consists of a Transmission Control Protocol/Internet Protocol client and server. The server opens a socket (i.e., a mechanism for creating a virtual connection between processes) on the host machine and connects to the client. The Installation, Force & Scenario, Results Display, and Export Manager interact with the server; the discrete-event simulation module interacts with the client; and the server completes a request from the client to make all connections. The type of communication TRANSCAP requires is well suited for socket use. The modules execute on the same machine and the messages are simple data strings, so a socket provides an easy and quick means of communication.

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 & Scenario Manager must also be changed so that it can create installations and scenarios; the discrete-event simulation 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 other 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 discrete-event simulation modules could simultaneously execute multiple scenarios. The pre-processing modules could be distributed via the Internet, and the discrete-event simulation could become a server, thus allowing Web-based simulations. Having separate processors or machines could also assist in 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 discrete-event simulation is written in MODSIM III. The Installation, Force & Scenario, Scenario Report, Results Display, and Export 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. A discussion of these three languages follows.

MODSIM III

A discrete-event simulation requires many constructs that most general-purpose programming languages do not provide. MODSIM III has been specifically created for object-oriented, process-based, discrete-event simulation. Its objects are inherently capable of synchronous and asynchronous actions. Therefore, this language is well suited for developing the discrete-event simulation module.

TRANSCAP uses object-oriented programming because of 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.

Of all of the object-oriented programming languages, MODSIM III features the most complete framework of vendor-supplied classes for discrete-event simulation programming. When using MODSIM III, there is almost no need to write statistical collection or discrete-event simulation facilitation source code.

Java

Java is an object-oriented language with advanced features like garbage collection, built-in security and memory robustness, extensive error detection and handling, and GUI application/programming interfaces. These qualities make it well suited for GUI development. The pre- and post-processing modules encapsulate almost all of the user's interaction with the simulation, and thus extensively utilize GUIs.

Java also provides support for network communication, making it appropriate for the server portion of the Inter-process Communication Manager. It is also a multi-threaded language that can take advantage of multiple processors or machines and could help enable a Web-based simulation.

C

C is a low-level, general-purpose language that provides socket libraries. C is used in TRANSCAP because MODSIM III did not provide socket libraries when the client portion of the Inter-process Communication Manager was written. Also, MODSIM III can be extended by an interface with C code. Therefore, the client was written in C.

REUSABLE DEPLOYMENT CLASS FRAMEWORK

The reusable deployment class framework was designed for TRANSCAP and has been reused in other MTMCTEA and Argonne deployment simulations (Burke *et al.* 1999). This framework, called 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 AMSO has officially adopted EXHORT's Transportation Class Hierarchy as a standardized code structure for object-oriented deployment simulation to ensure meaningful data exchanges.

This reusable deployment class framework was intended for any deployment, but extended for TRANSCAP to develop military fort-specific subclasses. These subclasses contain logic for specific interactions between clients, servers, and areas.

An example of how the reusable deployment class framework can be 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. When the vehicle is loaded, its current status is then updated. 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) to load 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 based on 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 is subclassed for a specific situation at a fort.

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 discrete-event simulation, simplifying the design and development, providing data abstraction, allowing selection of an appropriate language for each module, and providing greater freedom for future growth. The Interprocess 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 class 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

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 Graphical User Interface. 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.

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 and ACM.

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 University, 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 has 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 decision-makers.