INTEGRATED MODEL DEVELOPMENT ENVIRONMENT

Patrick K. Clark
TASC, Fairbom, Ohio

Captain Douglas A. Popken
AFHRL/LRL, WPAFB, Ohio

Airbase Logistics simulation analyses have historically been done using large monolithic software models, which are not well documented in terms of algorithms used, have no user friendly interfaces, provide little in the way of postprocessing and graphical display of simulation results, and allow limited flexibility to modify the algorithms. The Integrated Model Development Environment (IMDE) being developed by TASC through AFHRL/LRL sponsorship will address these problems, using object-oriented design and rapid prototyping philosophies. The environment is being written in C++, the simulation elements are being written in MODSIM II™ and stored using the VERSANT™ object-oriented database management system (OODBMS), and the user interface is being iteratively designed using devGuide™ from Sun Microsystems.

Simulation in all domains has long been at best a grey art, although notable exceptions exist in the communications and computer design areas. Many simulation models are originally developed for short term use, but are then evolved ad nauseum with little regard for maintaining a cohesive user interface or modular program structure. This results in the problems listed in the abstract, which can further lead to a lack of knowledge of what the model's capabilities actually are, as well as a propensity to abuse both input and output data. In absence of such abuse, there remains the large amount of time that can be spent in setting up input data or analyzing output data. In many cases the analyst must be both airbase logistics expert (in our case) and computer programmer, to decipher what exactly he can expect from the model, since the actual software code is often the only accurate documentation of the model's contents. This often translates in reality to more than one person being needed for any analysis, with the additional communication opportunities that usually presents.

The Integrated Model Development Environment (IMDE) is aimed at improving the analyst's productivity by providing him with tools to complete the entire simulation analysis, from model preparation, to simulation execution, through data analysis. These stages of the simulation process are reflected in the IMDE architecture (Figure 1) as the Preprocessor, Simulator, and Postprocessor. The Preprocessor constructs, configures, and documents simulation models or model parts. The Simulator manages execution of the model and collects desired raw statistical data. The Postprocessor analyzes raw data generated from the Simulator and produces requested reports. These major components interact with each other through data stored in either the Premodel or Project database (discussed below). The actual database access and component selection functions are resident in the Kernel Model Development Environment (KMDE), which localizes any changes to the environment required for database upgrades, operating system revisions, etc. All components will be integrated by a standards-based OpenWindows user interface.

Figure 1. IMDE Architecture
The Premodel database is used to store model parts, while the Project database contains simulation experimental frames that have been constructed by selective retrieval of parts from the Premodel database. The Project database also contains both raw and processed data associated with the relevant experimental frame. The Assistance database (not pictured), will provide detailed help and tutorial information on all IMDE commands.

In order to provide the most flexibility in both the environment and actual simulation models, both will be constructed using object-oriented programming languages. The benefits we expect to obtain from using this technology address several of the complaints regarding current models. Code interface protocols are strongly enforced compared to earlier languages, allowing a high degree of information hiding between loosely coupled program entities. This encapsulation of data with the operations that use it will localize the code that a software maintainer must examine for changes to that data. Software maintenance and modifications should become much easier as a result. In addition, different entities can have encapsulated procedures with the same names. For example, this feature, known as polymorphism, allows F-15 and F-111 model objects to both "own" possibly different procedures named "takeoff". Encapsulation and polymorphism combined with run-time binding allow us to interchange different parts into the model very quickly. To keep track of the large number of "plug-and-play" parts that will have to be developed to achieve flexibility and maintain the fidelity level of current logistics models, each part will be stored in the Premodel database with attributes shown in Figure 2. The parts are most naturally arranged using an OODBMS, due to the close parallel to the programming languages being used and the complexity of the data structures being stored. Complete simulation experiments will be equally complex, and will be stored as objects in the Project database with attributes as shown in Figure 3.

In recognition of the different types of skills that may be possessed by simulation modelers, we are designing four user levels into IMDE: Utility, Developer, Analyst, and User. This differentiation exists primarily in the Preprocessor component of the environment.

The Utility level user represents the analyst's traditional computer programmer sidekick. He will be responsible for initial coding of class protocols (Booch, 1991) for the specific simulation study required. The set of class protocols defines all the kinds of entities which will exist in the simulation, as well as what services are available to be used outside a specific entity (i.e. an "aircraft" entity may have a Built-In Test (BIT) procedure that can be called by a "maintenance" entity). The Utility user will also provide system management, database and language upgrades, and tool improvements for the IMDE.

The Developer level will allow for the development of new classes of entities that conform to existing class protocol definitions stored in the database by the Utility modeler. A Developer will be able to take a rudimentary "aircraft" class from the Premodel database and modify the algorithms of the "aircraft"'s services; then store the new class into the database as an F-15, B-1B, etc. These classes, both generic (aircraft) and specific (F-15), will then...
be available as reusable model parts in the Premodel database, and can be combined together to form "custom" simulations. The Premodel database thus becomes a library of useful parts that the modeler can use to quickly bring together a simulation model of his current, perhaps urgent, problem. This provides a high degree of flexibility not available in current models which have no separable reusable parts.

The Analyst user of IMDE will be able to select different model parts from the Premodel database and bring them together to form a complete executable simulation. As part of this process he will be able to select which statistics he wants to have collected on different variables of interest. The combination of a set of selected Premodel database classes becomes the basis of a Project. The Project database consists of a library of these constructed models, augmented with their respective input sets, documentation, and raw and processed output data sets. Figure 4 shows conceptually how the Projects are composed of parts from the Premodel database.

Figure 4. Project and Premodel database interaction

The User level is designed to allow the analysis of problems that have already been defined in terms of the types of model parts present. By selecting from such existing Projects in the database, a User will concentrate on looking at the effects of parameter variations to a model. For instance, by changing the numbers of spare parts available, or by varying the number and skill level of personnel in an avionics repair shop, the User will be able to do fairly extensive analyses, while still being guided by the modeling environment. The User will also be able to document input/output data sets.

The four level concept for IMDE users discussed above attempts to address the total simulation process, which includes personnel with varying mixes of requirements for domain knowledge and model implementation knowledge. The ladder of interaction between these levels is illustrated in Figure 5. Ongoing discussions and prototype evaluations with potential users of IMDE may validate the four level model or may suggest combining levels to reflect the reality of the modeling personnel structure. Regardless of what functionality eventually resides in each level, the capabilities for the design and analysis of much more flexible models will exist in IMDE.

Figure 5. IMDE Multi-level user concept

IMDE is being designed as a tool to help different types of people improve the quality of their analysis and shorten the time required to generate recommendations to decision makers. Our specific problem domain of concern is airbase logistics analysis, but the IMDE will be generic in the sense that any domain of simulation classes can be used to populate the Premodel database, and in turn to construct Projects for that domain. The object-oriented design and programming concepts are the keystone that makes such genericity possible. We feel these concepts are
essential in all areas of computer modeling to control the software complexity crisis we are facing in building larger software simulation models.

References:

