Proceedings of the 1999 Winter Simulation Conference
P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, eds.

SDI INDUSTRY PRO: SIMULATION FOR ENTERPRISE-WIDE PROBLEM SOLVING

Andrew J. Siprelle
David J. Parsons
Richard A. Phelps

Simulation Dynamics, Inc.
416 High Street
Maryville, TN 37804-5836, U.S.A.

ABSTRACT

SDI Industry Pro is a versatile, high-level simulation toolset for solving problems of whole enterprises. It adds important capabilities to an existing simulation package, Extend™, which provides a robust simulation architecture and a wealth of existing building blocks. SDI Industry Pro provides the ability to build multi-level models that address everything from manufacturing lines to entire plants, and supply/distribution chains.

1 EXTEND BACKGROUND

SDI Industry's underlying simulation engine, Extend, is an iconic block-based, graphical, general purpose simulation program developed by Imagine That, Inc. Blocks are connected to build models, and groups of blocks can be combined into hierarchical blocks (H-blocks) with unlimited layers of nesting (Diamond, 1993). Because of Extend's ease of use and attractive price, it has grown the market for a new generation of simulation users.

Extend provides a variety of iconic blocks, in files called libraries. The primary libraries are the Generic, Discrete Event, and Manufacturing. Blocks from the Discrete Event and Manufacturing libraries conform to a standard discrete event framework. Application of these blocks is easy to learn for simulation model builders who have become accustomed to items (or entities or orders), attributes, events, queues, delays, batching, and resources. The Generic library provides blocks, which can best be described as "nuts and bolts.” These are useful for calculations, and by extension, general control issues.

There are important benefits and considerations inherent in a simulation environment, which enables a block approach for building models. The genesis of SDI Industry has been a decade-long exploration of the block approach, its ramifications, and its opportunities. Using a block approach can result in models that are transparent and modular. By "transparent" we mean that models expose critical details about the way they are built, using intuitive icons. For example, in models built using templates of H-blocks (SDI Industry), it is possible to drill down within the hierarchy of a department to simple blocks that are hooked together to perform a logical function (e.g., tank sequence control). While the model is running, animation can be activated so that these blocks can update colors, levels, or values to indicate their state.

By "modular" we mean that blocks that handle well-defined tasks can be added to the model for specific embellishments. By constructing and using H-blocks, higher-level modules and templates can be used to quickly assemble models. A model comprised of H-blocks can be split up into multiple models, one for each H-block, and the H-blocks can be reused in new models.

On the other end of the spectrum, and by way of "extending Extend" through the coding of new blocks, it is possible to build models that are not transparent or modular; i.e., they are monolithic. For example, models have been built that are "one-block wonders"; i.e., the whole model is coded in one block. The "one-block wonder" typically includes a dialog that presents a dizzying collection of choices, all of which must be considered by the model builder. Outside of the ability to refer to the source code, such blocks are not considered very transparent. Unless the model user is also a programmer, such blocks are not very reusable. Beyond defeating the whole idea of a block-based system, the one-block wonder is not a desirable design approach for building simulations in general, let alone for building new blocks in Extend.

A logical system of blocks enables great opportunity to develop models progressively and with varied focus. It is undesirable for the model builder (or user) to have to commit to levels of detail that are unnecessary to the problem at hand (e.g., as in the dizzying dialog described above). Block modularity allows for the express construction of simple “sketch” models that can be expanded into greater detail, which matches subsequent stages of investigation and funding. The block approach also allows for models to be developed to varied degrees of detail which reflect interest in particular issues.

SDI Industry is the culmination of years of experience building Extend models and systems of Extend blocks to address complex problems. At the same time, it is a triumph of applying a block approach to building models and modeling systems that are transparent, modular, and reusable. SDI Industry is extremely versatile because of its modular approach and its compatibility with other Extend modeling tools. To appreciate the origin of this versatility, we will explore Extend's block development capability.

2 BLOCK DEVELOPMENT AND MODULARITY

Underlying Extend's ease of use is a powerful toolset and architecture for the development of new blocks. Extend provides a built-in block development language and compiler. The block development language is a subset of C called ModL. ModL supports standard C syntax, and adds a host of simulation- and statistics- specific calls and variables. ModL also provides calls that deal with blocks and H-Blocks and the basic nature of a system of interconnected blocks. All of Extend's blocks are built using ModL, and all of the blocks are sold with open source code. The code for implementing simulation logic goes extremely deep, while the user interface implementation is high level. For example, all details of event posting and handling are open. The block icon, its connectors, the block's dialog tabs and items are all created using high level visual tools. The bottom line is that the block developer can concentrate primarily on logic, and not on user interface issues.

Using ModL, it is possible to modify existing blocks supplied by Imagine That. It is also possible to build new blocks from scratch. If one builds blocks from scratch, it is possible to either conform to Imagine That's implementations or diverge entirely. In fact, whole block architectures have been created in Extend which are new and completely different than the ones supplied with the product. For example, a system of blocks has been built for simulating systems of airflow (e.g., Spray Dryers), with blocks to represent fans, ducts, and splitters. Another system of blocks was built to simulate cell phone nodes, for building cell phone network models.

Extend has a unique block message-based simulation architecture (Lamperti and Krahl, 1997), which allows block developers to build truly modular blocks and for model builders to use them. Using ModL, blocks can be programmed to initiate and/or respond to messages that can be transmitted through connections or system level "through the air” message calls. Thus, each block behaves as a kind of "micro simulator" which responds to messages, and takes action. Systems of blocks can be built to conform to messaging rules. Extend's own discrete event blocks employ a sophisticated messaging system, which ensures that models will run even when blocks are connected in an almost frivolous manner. An important benefit of a messaging system is that it is possible for models to update blocks on a "need to know" basis via the network of connections made by the model builder.

Extend's packaging of code in blocks and its messaging system encourages and enforces the building of modular blocks. This point cannot be overemphasized. Blocks provided by SDI Industry take advantage of this fact. They are fully compatible with blocks provided in Extend, so that model-builders can leverage all of the blocks provided in the standard Extend package.

There has been an on-going debate in the simulation community between sides advocating applications, which require code versus ones that do not. The author embraces important points made by both sides. Having explained the block approach to building simulation models, and the block development environment, the author wishes to make a case for block-based systems using standard languages (e.g. C, Pascal, or Basic) as the underlying development environment. It should be the goal of such systems to provide pre-built blocks that do not compel model builders to use code. These blocks should be well exercised across a wide population of users. Openness of the source code for blocks, and an ability to truly modify their behavior and to create new blocks is highly desirable in building industrial-strength models.

For model users and for companies that are using models for similar classes of problems across the corporation, the reusability inherent in a block approach is a clear benefit. However, the aspect of transparency (seeing how the model works) inherent in a model which employs a rational framework of blocks (and H-blocks) becomes increasingly appreciated by all serious model users, no matter what their level or proclivity for or against code. By employing a well-designed system of blocks, the model can be crafted to include only the detail necessary to address the problem at hand.

3 SDI INDUSTRY AND ITS ORIGINS

Simulation Dynamics was founded in 1990 based on the realization that simulation would be revolutionized by the advent of low-cost but powerful simulation packages such as Extend. As an affordable, easy-to-use iconic block building simulation package, Extend is unparalleled. However, for engineers who build real-world models, it is critical that a system's ease of use does not get in the way of flexibility. For engineers who are exposed to a wide spectrum of problems, Extend's flexibility is its truly unparalleled feature.

Through the experience of numerous projects done with Extend, Simulation Dynamics has dealt with recurring common issues. A constant goal has been to try to "extract the assets" of each project to understand where common themes exist. By looking for common themes, and refining the way in which systems of blocks and H-blocks are designed and used, SDI evolved a toolset with significant problem-solving potential.

Basic themes encountered have been in the areas of simulation model data management and model material representation. Early in the company's history, large corporations began to adopt Extend and apply it to problems across many different divisions. These companies began to address logistical problems associated with whole plants producing many products. For these corporations, many of their production systems include both batch and continuous mode processes. The processes make products in high volume usually at high speeds. The standard version of SDI Industry (non-Pro) addresses these themes.

More complex themes have been in the areas of production system representation, schedule definition, coordination of scheduled material runs through parallel and sequential systems within plants, and buffer management between systems. This work has extended to the area of automatic schedule generation within models, and supply/distribution chain modeling. SDI Industry Pro addresses these themes.

3.1 General Purpose Embedded Database

The standard version of SDI Industry adds to Extend the capabilities of a general-purpose embedded database, and discrete rate Flow blocks. The database addresses fundamental challenges associated with data management. In many projects there has been the need to manage large quantities of data, such as in problems where many products are produced, and where there are extensive relationships between tables of data. Many users want to manage model data using Excel. The model needs to dynamically access data in an extremely fast manner. Multiple blocks within the model need to deal with the same data such as for product or process attributes, so there needs to be a global model data structure supporting a decentralized block-based system.

The database capability in SDI Industry is comprised of an Excel add-in and a system of Extend blocks. All model inputs can be defined in an Excel workbook, with multiple tables defined on different sheets in the workbook. SDI Industry’s Visual Basic add-in provides a menu for aiding the user in defining and navigating tables. Model execution speed is addressed by embedding the data from Excel in a Database Manager block in Extend. This allows instantaneous memory-resident access to data, and the ability to ship models without the need for Excel. Database-aware blocks then address model-wide access. The Database Manager and all database-aware blocks know the complete structure of all tables and their data. Popup menus can be used to specify tables and fields within database-aware blocks. A table viewer is provided within Extend for viewing and editing data.

The idea of a having a centralized database within a model that uses decentralized blocks in its construction may seem like a contradiction. However, similar to the debate between centralized and decentralized government, great power and flexibility rests in the combination of the two. By use of the embedded database, models can be built that leverage the relational structure of data tables. This significantly clarifies and eases the modeling task. It also removes concern about the ability to handle detail or to scale the model, because the database is unconcerned about whether there are 10 products or 10,000. Use of the database also yields substantial benefits in simulation project management. As we shall see, the database (and the database functions of the SDI Tools API – Application Program Interface) is a platform extensively used in SDI Industry Pro. Database-aware blocks can also create tables as the model runs, aiding in reporting capability.

3.2 Discrete Rate Flow Blocks

The Flow blocks in SDI Industry represent a general breakthrough in the way that material is represented in simulation models. For representing high-volume or high-speed production (e.g., processing or packaging), or where there is the ability to view material flow in terms of rates (e.g., cash flow, document processing, facilities planning), the discrete rate flow blocks are a natural fit. This is a class of problem where the standard technique in discrete event modeling is aggregation, which results in inaccurate and slow models (Sturrock and Drake, 1996) (Also Siprelle, 1995). An example application of Flow blocks is in simulating the flow of materials through complete cereal plants, from batch and continuous processing operations through high-speed packaging lines.

The discrete rate technology employed by Flow blocks reduces the events in a model to only those that cause rate changes. When compared to models using the typical item/entity/order aggregating technique, models implemented using Flow have been shown to increase execution speeds by 100 times. This technology is not new- the author is familiar with a similar innovation that dates back to Pritsker's Packaging extension. What is new is the innovative way that Extend's block message-based system has been exploited, to provide a robust "erector set" of parts for assembling Flow systems.

Flow blocks ensure that changes in the potential rates in a network automatically propagate to other blocks on a "need to know" basis. This makes it possible to build arbitrary networks of Flow, with complex controls, and account for material flow and ensure mass balance. This also makes it possible for any Flow block in the network to know whether downstream or upstream conditions are responsible for the actual flow rate being less than the maximum potential flow rate. Flow blocks use this information to animate using a color scheme which dynamically presents whether they are satisfied, blocked, or starved as the model runs.

Beyond the classes of problems referred to as "tanks and pipes" or simply "high-speed" processes, discrete rate Flow has been used for interesting applications that are considered to be further afield. For example, Flow has been applied in a Call Center model, resulting in fantastic speed-up of execution. Flow has also been used to model the long-range planning of nuclear waste systems, where waste streams, facility learning curves and processing capabilities are best viewed as rates that adjust over long periods of time.

Flow blocks are compatible with other modeling systems based on Extend's event management executive (all standard libraries supplied in Extend). In SDI Industry, blocks are provided to transition material from items to flow and vice-versa. For example, material can flow continuously onto a palletizer until a full pallet is released as an item. Flow is naturally versatile in dealing with other kinds of material changes, such as when going from pounds of bulk material to containers. In the standard Extend package, Generic blocks are often hooked together to provide simple controllers. With Flow blocks it is no different. For example, one can connect decision and logic blocks to configure sequencing control for a bin. Examples of different kinds of bins, such as Lost-in-Weigh and Feed-When-Full, are provided in the package as H-block templates.

In using the database, database-aware blocks, Flow blocks, and blocks within Extend, a nice synergy can be achieved to develop models that deal with dynamically changing material flow. For example, schedules of product runs can be defined in database tables, and database-aware blocks can be used to look up new rates, package sizes, and material and equipment properties as these runs are scheduled.

The true versatility of the package comes about in the use of the entire toolset. Blocks from SDI Industry's libraries are designed to be used with blocks supplied within Extend. As we shall see, SDI Industry Pro rationalizes the use of Extend H-blocks, comprised of blocks from all libraries.

4 SDI INDUSTRY PRO

Complimenting the tools within SDI Industry, SDI Industry Pro adds libraries of Plant Builder, Schedule Generator, Supply Chain, and the SDI Tools API. The suite of tools in this package focuses on frameworks for building entire plants or warehouses, schedule definition and the managing of material flow from one end of a plant to the other, advanced resource definition, supply and distribution chains, and vertical model integration between plants and supply chains. In addition, the package is provided with open source code, and a high-level API (Application Programming Interface) for developing or modifying SDI Industry blocks.

4.1 Plant Builder Library and H-Block Levels

Blocks from the Plant Builder library provide a framework of controllers for building hierarchical blocks and templates adhering to the framework. The library also provides controllers for handling common tasks at levels within production systems and blocks to define and coordinate the running of schedules.

The most immediate contribution of SDI Industry Pro is a rational structure for defining H-blocks within Extend, and pre-built templates that adhere to this structure. Using the templates, it is possible to rapidly build an entire plant model that is comprised of multiple production systems with their own schedules, and intra-system buffers. One of the criticisms leveled at Extend in the past is that pre-built H-blocks are not supplied. Anyone can create libraries of H-blocks for use by others, but someone must still create them. Beyond supplying this, SDI Industry Pro's H-blocks are built to conform to a framework of defined levels of Equipment, Operation, System, and Plant. Each model level conforms to rules that guide its construction, the kind of data that is used at that level, and the appropriate process management blocks. These management blocks control the process at that level and provide an interface with levels above and below. The rigorous definition of levels and interfaces aids in model modularity. This modularity eases model navigation, understanding, H-block reuse, and the general practice of vertical model architectures and enterprise modeling.

While the use of these H-blocks greatly speeds up model building, it preserves the benefits of modularity inherent in the block approach. The model builder can create entire plants or single systems without the need to commit to detail within any given production system.

Therefore, the blocks provide the benefits of a high-level system without presenting the user with monolithic blocks. The upshot is that entire plants can be built as a "Shell Model," where high-level issues exist but lower-level detail is missing. In an actual project, a Shell Model of an entire plant was developed to immediately explore issues of schedule definition and management of work-in-process bins between processing and packaging. The initial database was then developed, and separate model builders created detailed processing and packaging system models.

Levels within SDI Industry are designed so that standardized controllers handle issues specific to each level. Equipment level controllers deal with the dynamics of processes such as failure, repair, and batch cycle control. Operation Level controllers manage changeovers and set process characteristics such as bin sizes or equipment rates, which may be based on material characteristics. System level controllers specify and read production/shift schedules, and coordinate operation startup and shutdown sequence as product changes occur. Plant level controllers manage distribution of product between systems, and coordinate their schedules.

Database tables are used extensively by controllers in SDI Industry Pro. Database tables can generally be defined in any way the user chooses. However, database tables configured for use by controllers must conform to certain rules. For example, changeover tables used by changeover controllers conform to a "From-To" matrix layout.

4.2 Schedule Definition and Coordination

The System level within SDI Industry is where staffing schedules, maintenance and production schedules are defined. A production schedule can be viewed as a plan, and what actually happens when the plan is executed can be different from the plan in semi-predictable ways. When the model runs, the System and Operation level controllers implement this plan in the model. The plan is specified by the model user in terms of production runs in a schedule in the model database. Simulation Dynamics has also developed tools to automatically generate schedules. We will defer the discussion of Schedule Generator blocks, and focus for now on schedule definition and execution.

The structure of production schedules allows for runs of different materials, flexible criteria for starting and ending each run, and also the definition of source runs, which will be explained through example. Runs of product can be individual batches, but they can also be continuous runs that are produced across multiple pieces of equipment within production systems. The System and Operation level controllers are designed to deal with material flow which is modeled appropriately either as items or as continuous flow using Flow blocks. In many systems modeled with SDI Industry Pro, there are batch and continuous operations within the same production system. At any point in time, material from a "run" can be present within the system model, in the form of batches (items/entities) in some areas, and levels of continuously flowing material (Flow) residing in bins or conveyors. Examples of allowable criteria for starting and stopping runs are clock time and a defined production goal, measured at a certain point in the system.

Users who are new to SDI Industry Pro are often surprised with the rigor required to define schedules for entire plants. In most cases, the surprise results from having to define relationships which are necessary for the model to know, but ones which are in the real system handled manually and on the fly by someone out in the plant. In virtually all cases, the user's work in preparing schedules according to the required structure has been a valuable experience in learning about their plant.

For example, an entire plant involving four stages of production has been modeled, where at each stage there are three or more production systems. Each system requires its own schedule. For some types of material, it is necessary to drain the plant of in-process inventory by the end of the week, while for others it is not. To model this plant using SDI Industry Pro, the schedules needed to have defined, in the appropriate runs, an associated source run. The optional source run number attribute in a production schedule identifies the upstream schedule run supplying material to the "user" run. By identifying source runs, the material can be emptied from upstream systems. The production schedules used by the real-world plant in this case were simple and vague compared to the schedules defined for the model.

This capability of "knitting schedules together," or dependent schedule runs, yields models that are capable of showing important real-world dynamics inherent in schedule coordination. As we will see, the rigor of schedule definition is exploited by the automatic Schedule Generator technology in SDI Industry Pro.

4.3 In-Plant Distribution Systems

In many plant configurations, there are groups of binbetween production systems. Sometimes referred to as in-process inventory, these areas can be viewed as product tank farms. SDI Industry Pro includes an H-block template and controllers for easily modeling these tank farms, in what it calls a Distribution System. This should not be confused with SDI Industry Pro's Supply/Distribution Chain architecture, which is a separate discussion.

Besides being islands of storage between production systems, a Distribution System can have a variable number of bins, with certain capacities, routing restrictions, and various material management rules. Because Distribution Systems stand between scheduled systems, they must handle the bookkeeping tasks necessary to coordinate the running of schedules from system to system.

In Simulation Dynamics' experience, there are plants, which have simple distribution systems, and others that resemble a veritable "museum of distribution.” The Distribution System in SDI Industry Pro has met the test of being easy to apply for the simple systems and capable of modeling the complex ones.

The goal of distribution is to get product from a set of one or more source systems to a set of one or more user systems. The distribution logistics must be one integrated strategy that can be made up of different combinations of source and user interactions with the distribution system. These are defined as four modes: Source System, Production Run (Batch), Material, and Bin.

For the sake of brevity, only the Production Run mode will be explained. In Production Run mode, each source production run is assigned one or more bins. Flow to these bins can only start after the bin is empty from the previous run. If no bin is available, the source system will be blocked until one becomes available. Note that in run mode, a system cannot dump material from a production run on top of material from a previous run, even if it is the same material and even if it is the same system that produced the previous run. This retains the integrity of the production batch. When a bin becomes full, flow will be routed to an empty bin if available. In Production Run mode, each bin is tagged with the production run that it contains, so that material can be routed to downstream systems requesting that batch. By configuring Distribution Systems with one of the aforementioned distribution models, an endless variety of plant configurations can be built.

4.3 Schedule Generator Library

In many projects, customers have wanted to have schedules automatically generated, based on a model of consumption, forecast demand, current inventory, identified constraints, inventory targets, reorder points, and specific schedule generation heuristics. Using the SDI Tools API, custom schedule generators have been built according to specific needs. This capability allows customers to study a myriad of issues, ranging from the effect of variability and seasonality in consumption, to the identification of one or more parts of the plant as bottlenecks.

By using a Schedule Generator within a simulation model, the user can validate and test heuristics and policies currently in place. These policies can easily be tested against other candidate heuristics with the simulation model to determine their relative effectiveness.

A generic version of Schedule Generator blocks is now available within SDI Industry Pro. These blocks can be connected together to create a system that successively generates schedules for all systems in a plant model. The blocks works in concert with Supply Chain blocks to allow a flexible definition of finished goods consumption, and ordering and inventory replenishment from model production. The system of Schedule Generator blocks can be configured to consolidate orders and create a new schedule for any horizon of time. The blocks create system schedules successively using an approach based on the Theory of Constraints. Using these blocks, any type of system in the plant model can be identified as a constraint, so that it is scheduled first based on consolidated requirements. This schedule is then rolled forward or backward to subordinate upstream and downstream systems in the order they are connected. The framework addresses the needs of users who have wanted to test the effect of scheduling assuming alternative constraints.

4.4 Supply Chain Library

The purpose of a supply chain simulation is to determine the impact of changing demand, logistic decisions, production policies, and supplier performance on the reliability, velocity, cost and capacity of the system. Supply chains interact dynamically with market uncertainties, plant operations, and supplier performance.

Through a variety of supply and distribution modeling projects, Simulation Dynamics has evolved a versatile system of blocks for modeling supply chains and/or distribution chains in the same model. In many of these projects, there has been an interest in modeling a production plant within the context of a supply or distribution network. Vertically integrating a supply chain model with its production plant counterpart can produce valuable information about how improvements can be made to both. The results of running the two models in an integrated overall model can be significantly different than running the models separately.

Blocks in SDI Industry Pro's Supply Chain library are designed in accordance with the Extend vision of modularity, so they allow progressive development. These blocks are compatible with other Extend tools, so that models can be comprised of blocks from Extend+Mfg or SDI Industry Pro. With the Supply Chain blocks, inventory levels, unassigned and assigned orders, and shipments can be passed between block connectors. The user can easily link supply nodes to the raw materials of SDI Industry Pro plant models, and plant finished goods inventory can be linked to downstream distribution nodes.

The basic supply chain block types in the Supply Chain library employ a scope definition, which allows a narrow or broad focus on locations and/or materials. For example, the user can easily choose to configure an ordering process to apply to one product or to a group of products. This capability is built using the embedded database, so that the process is instantly scaleable to large numbers of products and/or locations.

The basic Supply Chain blocks characterize consumption, ordering, order assignment, order filling, and routing. Each block can be configured to work using default rules that involve very little setup. However, each step is clearly segmented and can be overridden, either by the user's own custom block or by a sub-model. For example, the user can install an event in the model to handle a consumption process for a certain product, and that event can be handled by a small, embedded sub-model implemented as a discrete event or Flow network.

Supply Chain blocks provide a standard yet flexible way to deal with resources such as containers, material handling equipment, and vehicles. Logistic studies will vary in their interest in these elements and should not require custom add-ons divorced from the basic data structure of the model. Critical resources such as containers, trucks, ships, or other entities are treated as items with their own inventories and logistic rules. The hierarchical shipment structure allows for any number of "Russian Dolls" to be contained within one another. In addition, resources and materials can be shipped according to highly flexible routings which are defined in the database.

At the core of SDI Industry Supply Chain is a clear definition of the basic elements that make up a supply or distribution system. The goal of the conceptual structure reflected in this definition of basic elements is application to a wide range of potentially interrelated logistic problems. The embedded database is used to characterize these basic elements of locations, items, inventories, and shipments.

4.6 SDI Tools API

With the exclusion of the Database Manager and Supply Chain Manager blocks, all blocks within SDI Industry Pro are supplied with open source code. The SDI Tools API represents an extension of the ModL procedure and function set. Over 50 procedures and functions are provided. The purpose of the API is to allow for the development of new database-aware blocks and supply chain blocks, according to the database and supply chain architectures within SDI Industry.

Data-aware blocks are typically built to provide general data access, to work with experimental data more flexibly, to create custom reports, or even to build custom Schedule Generator blocks. The database function calls allow the programmer to dynamically add or delete tables, fields, and records as the model runs.

The Supply Chain blocks are programmed using a set of high level extensions to the ModL programming language. These extensions are made available in the SDI Industry Tools API to allow users to develop new supply chain blocks. A key feature of the programming language is the separation of the triggering and execution of events. Procedure events occur as a result of 'doProcedure' or 'doDelayedProcedure' calls. These calls can be executed in any block, not necessarily the block, which will execute the procedure. For example, doAssignOrder(order) will cause the assignOrder procedure to occur in the block registered to handle it. The doDelayedAssignOrder(order) command will cause the order to be assigned at the specified time in the future. This separation allows the programmer to insert preprocessing steps between the triggering of a procedure and its execution. Preprocessing could include aborting of the procedure if certain conditions are not met.

5 CONCLUSIONS

Simulation Dynamics has delivered powerful solutions to customers in a wide spectrum of challenging simulation modeling projects for 10 years. The tool of choice for ultimate usability and flexibility has been the Extend simulation platform.

The block architecture and development system has inspired Simulation Dynamics to evolve the state of the art in modular model building, and to refine the system of blocks for addressing classes of problems in a general and flexible way. SDI Industry Pro provides a suite of tools that allow rapid modeling of systems, from simple lines to plants and entire enterprises. For enterprise modeling, the tools can be used to assemble vertical model architectures that have varied degrees of detail as necessitated by the problem.

SDI Industry's suite of tools has significant problem-solving potential, and they can be exploited by practitioners of simulation at any level of interest or proficiency.

REFERENCES

Diamond, B. 1993. Extend: A Library-Based, Hierarchical, Multi-Domain Modeling System. In *Proceedings of the 1993 Winter Simulation Conference*, ed. G. W. Evans, M. Mollaghasemi, E. C. Russell, and W. C. Biles, 240-248. IEEE, Piscataway, NJ.

Krahl, D. 1994. An Introduction to Extend. In *Proceedings of the 1994 Winter Simulation Conference*, ed. J. D. Tew, S. Manivannan, D. A. Sadowski, and A. F. Seila, 538-545. IEEE, Piscataway, NJ.

Krahl, D. and Lamperti, J. S. 1997, A Message-Based Discrete Event Simulation Architecture. In *Proceedings of the 1997 Winter Simulation Conference*, ed. S. Andradottir, K. J. Healy, D. H. Withers, and B. L. Nelson, 1361-1367. IEEE, Piscataway, NJ.

Siprelle, A. J. and Parsons, D. J. 1995. Modeling a Bulk Manufacturing System Using Extend. In *Proceedings of the 1995 Winter Simulation Conference*, ed. J. D. Tew, S. Manivannan, D. A. Sadowski, and A. F. Seila, 813-817. IEEE, Piscataway, NJ.

Sturrock, D. T. and Drake, G. R. 1996. Simulation for High-Speed Processing. In *Proceedings of the 1996 Winter Simulation Conference,* ed. J. M. Charnes, D. J. Morrice, D. T. Brunner, and J. J. Swain, 432-436. IEEE, Piscataway, NJ.

AUTHOR BIOGRAPHIES

ANDREW J. SIPRELLE is President and founder of Simulation Dynamics, Inc., a firm that provides simulation consulting, training, and custom models. Mr. Siprelle's industry experience includes creation and analysis of models for strategic and capacity planning, market analysis, and the application of industrial statistics. Applications of simulation have been in the areas of business, manufacturing, government and service operations. Before starting Simulation Dynamics, Inc. he worked at the Aluminum Company of America. He received his B. S. in Industrial Engineering and Operations Research from Virginia Polytechnic Institute.

RICHARD A. PHELPS is the Projects Manager at Simulation Dynamics. Mr. Phelps' industrial experience prior to joining SDI includes time in the cutting tool and aluminum industries. He received his B. S. in Industrial and Systems Engineering from the Georgia Institute of Technology where he is a member of the Council of Outstanding Young Alumni.

DAVID J. PARSONS is a principal of Simulation Dynamics. His experience with simulation began in 1965 with experiments in the use of natural selection algorithms to evolve architectural designs. During the 1980's he designed, built and operated several dairy-processing plants using simulation of key systems as an integral tool for design, value engineering, and trouble shooting. Mr. Parsons received a B. A. from Harvard College and a Master of Architecture degree from the Harvard School of Design.