



PlanetTogether

*Stellar Performance in Planning and
Scheduling for Multi-Plant Manufacturers*

Selecting Scheduling Software



Selecting Scheduling Software

The shop floor scheduling problem has been around for a long time. The research community started publishing articles in the 1950s, with a flurry of stories about it appearing in the '60s and '70s. The first book on industrial scheduling was published in 1963, and the first book on the theory of scheduling was published in 1967. Although there was a vast amount of research devoted to this problem, it soon became apparent that the general problem would not be solved easily. The field of complexity theory emerged in the late '70s and early '80s, which basically proved that it was unlikely the general scheduling problem would ever be solved.

The late 1980s produced the first commercial software packages aimed at providing a good solution to the shop floor scheduling problem. Early scheduling systems were originally developed because of the infinite-capacity assumption used by many planning systems. These early systems were often referred to as finite-capacity scheduling (FCS) systems, and most of them used simulation concepts to develop a feasible solution. Although they were capable of developing good solutions, they often required a significant amount of time to generate that solution. The time problem largely has been resolved with the recent advances in computer technology. Today's computers are faster, cheaper, and allow much larger problems to be solved. Solutions often can be generated in only a few minutes, even on personal computers.

Advanced Planning and Scheduling Systems

With the strides made in computing technologies, a large number of software packages then emerged in the '90s that became collectively known as Advanced Planning and Scheduling (APS) systems. These systems were a natural extension to manufacturing resource planning (MRP II), enterprise resource planning (ERP), and manufacturing execution systems (MES).

These early APS systems were far from homogenous; they included a wide range of software applying distinctly different solution approaches and often focused on solving different problems. Beyond functionality, these systems also ranged in price from less than \$50,000 to several million. Despite their differences, this market has grown at a rate of 70 percent, and since the early 2000's, several attempts have been made to categorize the different approaches, but there is not an accepted standard as of yet. The proposed categories have been based on three things: the solution time frame, the approaches, and the focus of the solution. From the user's perspective, the solution focus provides the most meaningful sub-categories that are increasingly prevalent today: supply chain; multi-plant; and plant scheduling. The scheduling survey included in this issue of IIE Solutions focuses on the latter two sub-categories.



PlanetTogether

Supply chain planning software is the most expensive and focuses on developing a comprehensive plan for the entire supply chain. These systems often include many of the features traditionally found in MRP II and ERP packages. They are capable of developing plans for multiple facilities and suppliers, as well as the logistics. The primary goal is to determine what to make and where to make it. The emphasis is placed on developing a cost-effective plan, not the scheduling of individual operations. Some of these systems do not even include detailed shop floor scheduling- or at least not as part of the basic software offering.

Plant scheduling software is at the other end of the spectrum, often being the least expensive and focusing on developing detailed operation or job schedules. This software typically assumes that a plan has been developed, often coming in the form of order releases from an MRP system. The more sophisticated systems also consider material and component requirements.

Multiplant scheduling software falls in the middle and can include a wide range of software products. Although the primary focus is on developing detailed schedules, these packages are capable of allocating orders over multiple facilities, as well as developing schedules. These software packages often include some of the basic features normally found in MRP systems.

Developing a Schedule

Several basic methods are used by all packages for developing shop floor schedules. Again, there is no accepted standard for classifying scheduling methods. A simple classification yields four methods: job (algorithmic), resource based, event (simulation), and optimization.

Job scheduling is the simplest method. In some ways, it mimics the method that a scheduler might use to develop a schedule using a traditional magnetic board. It begins by ordering the set of jobs to be scheduled using a selection rule. A large number of rules are often available, but they tend to be simple, static rules (e.g., due date, release date, highest job priority). In most packages, complex rules may be available or additional rules may be added. All operations of the first job are then scheduled. This procedure is repeated for the remaining jobs. These methods tend to be very fast, but they can produce schedules with gaps and large job cycle times. They are also not suitable if special constraints exist (e.g., sequence-dependent setups).

Resource-based scheduling is often loosely based on the theory of constraints. The idea is to first schedule the operations that require the defined bottleneck resources in the system. The remaining operations are then scheduled on the nonbottleneck resources using a combination of backward and forward schedule techniques. Although the theory of constraints is a well-known concept, the actual methods used to develop a schedule can vary greatly. These methods can produce good schedules if there are true bottlenecks in the system, but the solution time is greatly dependent on the package.

Event scheduling employs simulation concepts to develop a schedule. Basically, it starts at the current time and tries to schedule all possible operations. It then steps forward in time until



conditions change to allow additional operations to be scheduled. The method for selecting which operations are scheduled first can be resource-dependent and dynamic (e.g., critical ratio, minimize setup). Most packages also allow special user rules to be developed. This method tends to produce schedules with fewer gaps, but the solution time can be large for complicated systems. However, it can provide good solutions to systems where sequence-dependent setups or other special constraints are present.

Optimization scheduling methods are provided by some of the packages. These methods are generally based on some type of search procedure, with the quality of the solution directly dependent on the amount of solution time allowed and the quality of the search procedure. Although these methods do not guarantee an optimal solution and require a long duration to generate a good solution, they are optimal seeking and work best for very complicated scheduling problems. Furthermore, they tend not to work well for systems that have day-to-day changes.

Implementing a Scheduling Package

The service ratio for implementation is typically between 0.5 and 1.5, depending on the package and the complexity of the installation. Some vendors require that they perform the installation, where others allow the option of user installation (with some training). Implementation can be separated into four activities: creation of a model, integration, customization, and training.

The first step is to develop a model of the manufacturing system using the constructs provided by the package. Some packages require the user to conform to the model that is provided, while others contain a wide range of constructs that can be selected. The package should be capable of modeling all key system constraints accurately. Many even allow user names to be used during the scheduling process.

As soon as the scheduling model has been defined, the package can be integrated with other existing systems. This integration requires that job, system status, and calendar data be passed to the package; the resulting detailed schedule data be passed back to existing systems; and periodic updates of actual performance from the shop floor be passed to the package. This transfer of data can perform automatically or at the request of the user. The transfer methods can vary greatly but typically take one of several common forms. The most common is the transfer of ASC II data, which requires that a code be added to all packages to either write out or read in the required data. Many packages now provide the means to link applications directly using ODBC technology, which normally requires the least amount of time and allows for automatic data transfer. Of the four implementation activities, integration normally requires the greatest amount of time and is critical to the success.

Typically, some form of customization is required for all applications. This can take the form of nonstandard reports, development of special user rules, or the addition of nonstandard features. Many packages now use second-party reporting software, which allows the easy development of specialized user reports. The ability to include special rules can be critical to the success if the manufacturing system has unusual constraints. Non-standard features should only be added if they do not currently exist in available packages or if software costs restrict



the options.

The final step is the training of users and the individuals who will be responsible for maintaining the system. Training can take from several hours to a week, depending on the user and the complexity of the system. Generally, it is recommended that the existing scheduling system be used, with the new system running in parallel, until user confidence has been established.

Selecting a Scheduling Package

The selection of a scheduling package should be based on five criteria: features, integration, technology, platform, and price. The most important factors are features, integration, and technology. Ideally, the selected package should have standard features that are capable of capturing all key system constraints accurately. The package should allow for easy, seamless integration with existing systems. This is greatly aided by packages that include the latest software integration technology. Computer platform and price should only serve as constraints in the selection process. With the currently available networking capabilities, platform is typically not an issue. In fact, most scheduling systems are now being installed and run on personal computers. Price is always a constraint, but with the wide range of packages on the market, the choice usually includes several alternatives. However, the buyer can't lose sight of the fact that the installation costs (both the service costs and the internal costs if special routines need to be created in-house) must be included in the total price.

The best way to select a scheduling package is to develop a small set of data that includes all the key elements of your manufacturing system and have selected vendors create a demo using those data. This allows for a common base of comparison between competing packages.

Even a poorly chosen package can provide benefits; a properly chosen package that integrates well with other applications can provide significant benefits. A good scheduling package will provide a window into the future that will allow you to provide accurate completion dates, anticipate problems, and play the "What if?" game when conditions suddenly change.

For further reading Conway, R.W., W.L. Maxwell, and L.W. Miller, *Theory of Scheduling*, Addison-Wesley, 1967. Kirchner, B., "Finite-Capacity Scheduling Methods," *APICS* The Performance Advantage, vol. 8, no. 8, August 1998. Muth, J.F., and G.L. Thompson, eds., *Industrial Scheduling*, Prentice-Hall, 1963. Parker, R.G., and R.L. Rardin, "An Overview of Complexity Theory in Discrete Optimization: Part 1, Concepts," *IIE Transactions*, vol. 14, no. 1, March 1982. Parker, R.G., and R.L. Rardin, "An Overview of Complexity Theory in Discrete Optimization: Part 2, Results and Implications," *IIE Transactions*, vol. 14, no. 2, June 1982. Turbide, D., "The Many Faces of APS," *APS* Advanced Planning and Scheduling, vol. 1, no. 1, June 1998. Randy Sadowski, Ph.D., is chief applications officer and director of university relations at Systems Modeling Corp. He has served on the faculty at Purdue University in the School of Industrial Engineering and at the University of Massachusetts. Sadowski received his bachelor's and master's degrees from Ohio University and his Ph.D. from Purdue. He is a senior member of the Institute of Industrial Engineers and was the founder of the IIE/SM student simulation competition. He currently directs that activity for Systems Modeling.

