
Ascent Technology, Inc.
Building 200
One Kendall Square
Cambridge, MA 02139-1589 USA
Telephone: +1.617.395.4800
email: sales@ascent.com
www.ascent.com

The Ascent WorkZone[®] workforce-management system



To Optimize Work Schedules, First Optimize the Scheduling System

It's hard to manually create work schedules that perfectly align costs, work rules, customer satisfaction, and worker satisfaction. Doing it with less-than-optimum software is even harder.

Airlines, airports, hospitals, casinos, and other labor-intensive businesses all face the problem of knowing which workers to deploy when, where, and for how long. Consistently staffing to peak levels wastes money and leaves workers idle during slower periods. But always staffing to normal levels causes worker shortages and upset customers during busy periods. In addition, different departments may have their peak, normal, and slow periods at different times. Other issues include seniority, vacations, shift preferences, mandatory rotations through work locations, balancing the ratio of full-time versus part-time, and variations in workload throughout a day, week, season, or year. Then, there is the weather to consider, plus any other disruptions, either planned or unplanned.

Because solving this workforce management problem is hard to do manually, organizations often seek a software solution that can, among other things, determine:

- How many workers are needed on each day and at each time of day
- The best start and end times for each shift
- The best work schedules to handle demand
- Work schedules that can be actively managed by workers and administrators
- Real-time work assignments based on current conditions
- The right balance between low costs and high customer and worker satisfaction.

Automating the solution, however, presents its own challenges. You wouldn't want, for example, to ask a team of developers to write a custom program that allocates workers based on just one specific set of work requirements. That would be slow and expensive to implement initially and to modify later whenever you needed to accommodate, say, a new work rule or business expansion.

Nor would you want a system you can buy off the shelf that delivers less-than-complete results. In that case, you pretty much have to redo the work manually to identify any gaps the solution missed and then fix them. That may take even longer than if you just solved the problem from scratch with no automation involved. That same duplication of effort applies every time—whether you are planning work periods, creating work schedules, making real-time shift assignments, or doing long-range scenario modeling (which is something beyond the scope of most less-than-complete solutions).

It's not just that the problem is inherently complex, which it is. To add value, any automated system itself must also be easy to use, fast, cost-efficient, and flexible enough to handle all the changes occurring in the workplace. It must fully capture, as input, every requirement that bears on worker schedules and, as output, it must generate schedules that are 100% accurate with no need for human tweaks.

No Tweaking

In fact, what organizations quickly discover when using a less-than-complete solution is that there is no such thing as a tweakable automated schedule. The system either produces a schedule you can trust right from the oven or one that is worse than worthless because of all the extra work needed to make the schedule useful. There is no middle ground.

Getting a trusted schedule on the back end means capturing absolutely everything on the front end that goes into the scheduling process. Even the slightest omission of some key elements can ripple through the entire schedule. Take something as seemingly trivial as Joe can't work after 2:00pm every Friday. Not capturing that rule correctly will, in all likelihood, randomly and occasionally place Joe on a Friday afternoon shift. Not only will that assignment not work; it also means moving other workers around to cover for Joe, which, in turn, affects those other workers' schedules (not to mention the tasks that need to be covered), all of which must be managed in the context of mandatory workplace rotations, preferences, seniority, and so forth. And, it also means finding a place for Joe to work his shift—causing more ripples, again in the context of all the preferences and work rules that may apply.

Not only must you make all these adjustments manually, which is a nightmare by itself, but you also have to spot the mistake in the first place. That means if you are the scheduler that you must remember all those rules in your head, and there may be hundreds, or thousands, or tens of thousands, of rules. Not remembering can lead to a vicious cycle of more time consuming and costly errors on top of other errors leading to still more errors.

No Custom Programming Either

Avoiding all these errors and wasted effort is why you would use an automated system in the first place. That requires a system robust enough to include all the specialty rules that may apply. One way to achieve that is for programmers to translate all the work rules into software code. That requires that someone (probably the vendor) to create a list of specifications from your HR policy manual and other sources, and write a custom program that performs to those specifications exactly.

There are several problems with the custom programming approach, including:

- **It takes a long time.** Collecting all the work rules, writing the specification, and then writing and debugging the software is a long process with numerous back-and-forth iterations.
- **Testing is extremely difficult.** Before the solution can be deployed, you will want to feed the system test data and evaluate the results to see if they are reasonable. The problem is that the custom code is a black box, meaning that it is very difficult to trace an error to the specific software code at fault. It could be a programming technical error; it could be a bad work rule; it could be a bad specification of the work rule; or it could be a bad translation of the specification into software code. It's also possible that even if the individual work rules are captured correctly that there is a problem with the program's overall logic—or any combination of factors causing the program to fail. The point is that finding and correcting errors—without introducing more errors into the code—is hard in large part because software code looks a lot different than the work rules on which it is based.
- **Software developers are not you.** If you hand a software developer a specification, then what you are asking for in return is software that does what the specification says it should do. That means any obvious problem in the specification itself will probably be overlooked and simply be written into the code. Had the developer, however, known what the specification was supposed to say, then the specification—and therefore the work rule implementation—would have been correct. And lot of obvious mistakes could be avoided.
- **Hardcoded rules are inflexible.** Organizations change all the time. Businesses expand. Labor contracts get renegotiated. Innovations affect the number and skills of workers needed. Products and services go out of style or come into style. New laws are passed, and so on. The problem with a hardcoded solution is that any change requires a software patch, which replaces part of the software program with updated code. Writing the new piece of software entails all the issues just described. It also causes unintended consequences as software modules often depend on other software modules in order to function properly. These interdependencies cause ripple effects, each of which must be identified and corrected before the revised program is ready to go into service.
- **Maintenance is very expensive.** Keeping the system current with new requirements is not only slow, difficult, and error-prone, it is also very expensive. That's because every time a change is required, the vendor has to write new code and debug the program—a process that incurs significant hourly programming charges. A better approach is when you can express the work rules yourself directly; and have the software implement them automatically for you with no custom programming or human tweaks required.

Who we are

Since our founding 25 years ago by members of the Massachusetts Institute of Technology Artificial Intelligence Laboratory, Ascent Technology has helped organizations deploy costly resources as efficiently, effectively, and economically as possible. Our highly trained and capable team of technologists, problem solvers, and solution designers has broad domain expertise and substantial experience in artificial intelligence, computer science and engineering, system design, mathematical optimization, operations research, and resource optimization, planning, scheduling, and management.

Don't Program, Configure!

A third approach is to use a system that incorporates an automated work rule modeler. The modeler avoids the manual tweaking involved with less-than-complete solutions and also avoids the manual coding and recoding that custom programs require. A work rule modeler consists of two parts:

- An easy-to-use interface that enables administrators to express all the rules in language that senior management, union management, and fellow schedulers understand
- An automatic translator that converts those entries into a program that executes the work rules efficiently for both fast computation and efficient workplace operation.

The problem that an automated work rule modeler solves is that human schedulers, as a rule, don't speak computer code and computers don't understand HR policies, labor agreements, or other work rule documents. To solve that problem, the modeler provides an alternative to both manual schedule writing and manual software writing. That alternative is to configure the system rather than program it. When you configure, you express your rules—all your rules—in English-like entries that you, the user, can understand. The system then interprets these entries and invokes the appropriate program logic that generates the optimum work schedules.

The program code is invisible to the user, and so, too, are the optimization and allocation algorithms the system uses to find the lowest cost and highest satisfaction solution within the prevailing rules and preferences.

The interface that captures the rules can be implemented in several ways, such as a form, a questionnaire, or a tree structure that allows for arranging statements in their logical order:

- 1) If a worker has seniority, give his or her shift preferences priority; otherwise don't give the worker's shift preferences priority
- 2) If the worker with priority has not worked more than 40 hours in the period, schedule the worker

With this interface, terms like "has seniority" and "more than 40 hours" could be selectable from drop-down menus.

The type of interface depends on the workforce management task. Forms, for example, may be the most efficient way to capture the information needed to define a workload (i.e., number of tasks, number of workers, and number of hours needed for each task). Near-term assignments, on the other hand, might be better served with a tree-like interface.

Scenarios Help You Plan Your Future

What matters is that, from the user's point of view, the system seems to program itself. Not only does this facilitate rapid and error-free schedules under the current rules, it also lets you easily build scenarios that help you see how those schedules will look going forward as rules change over time.

Multiple scenarios can reflect variations of rules through time, allowing you to see how different rule sets affect each schedule. Long bid lines, for example, that cover months or even a whole year may well cross multiple versions of a labor contract. When the system generates those schedules, it must consider those changes, juggling them through time as it works to generate a single schedule. Scenarios also let you evaluate different hypothetical work rule sets so you can decide which ones make the most sense for your bottom line, your customers, and your workers.

Building these scenarios would be impractical without a complete end-to-end solution—one that generates complete finished schedules directly from user-understandable input without tweaking schedules and without manual coding. That's the key measure of any automated workforce management system, whether for testing possible future scenarios or for generating the optimum schedule with which to allocate workers now. The optimum schedule perfectly balances costs versus service levels versus customer and worker satisfaction, given all work rules and worker preferences, regardless of how many workers, worker types, or work rules there are. The sooner you get that schedule, the sooner you can focus on what really matters—the work.

More information

To learn more about how Ascent Technology solutions can help you optimize your resources to greatest advantage, send email to sales@ascent.com or call our Sales and Marketing department at +1.617.395.4800.

ARIS, ARIS/AR, ARIS/AV, ARIS/BB, ARIS/CI, ARIS/CX, ARIS/FW, ARIS/GateView, ARIS/GM, ARIS/IQ, ARIS/LegGen, ARIS/PX, ARIS/SA, ARIS/SB, ARIS/SE, ARIS/SmartBase, ARIS/SmartBus, ARIS/SP, ARIS/Tow Panel, ARIS/WorkModel, ARIS/WorkNet, ARIS/WorkOptimize, ARIS/WorkPlan, ARIS/WorkRelay, ARIS/WorkTime, Ascent Technology, Inc. (stylized), Ascent WorkZone, Ascent WorkZone (stylized), GateKeeper, SmartAirline, SmartAirline Capacity Analyzer (stylized), Smartairline Operations Manager (stylized), SmartAirline WorkZone, SmartAirline WorkZone (stylized), SmartAirport, Smartairport.com, SmartAirport Capacity Analyzer, SmartAirport Capacity Analyzer (stylized), SmartAirport Information Manager, SmartAirport Information Manager (stylized), SmartAirport Operations, SmartAirport Operations Center, SmartAirport Operations Manager, SmartAirport Operations Manager (stylized), SmartAirport WorkZone, and SmartAirport WorkZone (stylized) are registered trademarks of Ascent Technology, Inc. ARIS/AR Display Board, ARIS/AR Turn Generator, ARIS/CA, ARIS/Reports, ARIS/SCR, Location editor, Reference editor, Resource editor, Rule editor, SmartAirline Capacity Analyzer, SmartAirline Operations Center, SmartAirline Operations Manager, User editor, Work schedule editor, and Worker editor are trademarks of Ascent Technology, Inc. This is not a complete list of all registered trademarks, trademarks, and service marks owned by Ascent Technology, Inc. Other company, product, and service names may be registered trademarks, trademarks, or service marks owned by other parties.