Scheduling for Cellular Manufacturing

Roman van der Krogt[1], James Little[1], Kenneth Pulliam[2], Sue Hanhilammi[2], and Yue Jin[3]

[1] Cork Constraint Computation Centre
Department of Computer Science
University College Cork, Cork, Ireland
`{roman,jlittle}@4c.ucc.ie`
[2] Alcatel-Lucent System Integration Center, Columbus, Ohio
[3] Bell Labs Research Center Ireland

Abstract. Alcatel-Lucent is a major player in the field of telecommunications. One of the products it offers to network operators is wireless infrastructure such as base stations. Such equipment is delivered in cabinets. These cabinets are packed with various pieces of electronics: filters, amplifiers, circuit packs, etc. The exact configuration of a cabinet is dependent upon the circumstances it is being placed in, and some 20 product groups can be distinguished. However, the variation in cabinets is large, even within one product group. For this reason, they are built to order.

In order to improve cost, yield and delivery performance, lean manufacturing concepts were applied to change the layout of the factory to one based on *cells*. These cells focus on improving manufacturing through standardised work, limited changeovers between product groups and better utilisation of test equipment. A key component in the implementation of these improvements is a system which schedules the cells to satisfy customer request dates in an efficient sequence.

This paper describes the transformation and the tool that was built to support the new method of operations. The implementation has achieved significant improvements in manufacturing interval, work in process inventory, first test yield, headcount, quality (i.e. fewer defects are found during the testing stage) and delivery performance. Although these benefits are mainly achieved because of the change to a cell layout, the scheduling tool is crucial in realising the full potential of it.

1 Introduction

Alcatel-Lucent is a major player in the field of telecommunications. One of the products they deliver to network operators is wireless infrastructure. A key component of wireless infrastructure is the base station. Such equipment is delivered in *cabinets*, an example of which is pictured in Figure 1.[1] Located between the

[1] Although the cabinet is only the outer casing of the equipment, this is the term the company uses to refer to the equipment. In this paper, we will therefore also use the term 'cabinet' to refer to the whole item.

antennae and the ground network, their function is to handle the signals the antennae receive and send. Depending on the model, their size is roughly that of a standard kitchen refrigerator and they are packed with various pieces of electronics: filters, amplifiers, circuit packs, etc. The exact configuration of a cabinet is dependent upon the circumstances it is being placed in: the type of network (e.g. CDMA or UMTS), the frequency (the GSM standard defines eight frequency bands, for example), physical location (inside or outside), network density (is it expected to handle a high or low volume of calls, what is the area the cabinet covers), etc. Some 20 product groups can initially be distinguished. However, as one can understand from the many aspects that are taken into consideration, the variation in cabinets is large, even within one product group. For this reason, they are built to order.

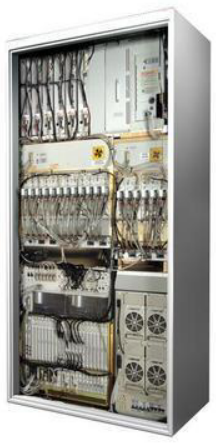


Fig. 1. A cabinet

The site that we worked with, the Systems Integration Center in Columbus, Ohio, produces several hundred cabinets per week on average. The production takes place in three stages: assembly, wiring and testing. The durations of each the stages depends on the particular product group of the cabinet.

Assembly. The first series of steps takes as input a partially pre-populated cabinet. This has certain basic features such as power supplies, cooling and the back plane. To this, they add the required amplifiers, filters, circuit packs and other hardware according to a predetermined schema. The cabinet passes through a number of stations. Each station is dedicated to one or more types of modules that are fitted into the cabinet.

Wiring. The second step involves physically interconnecting the hardware that was added during assembly. Each component has a number of input and output ports that have to be connected with a wire to the outputs and inputs of other modules, again according to a predefined schema.

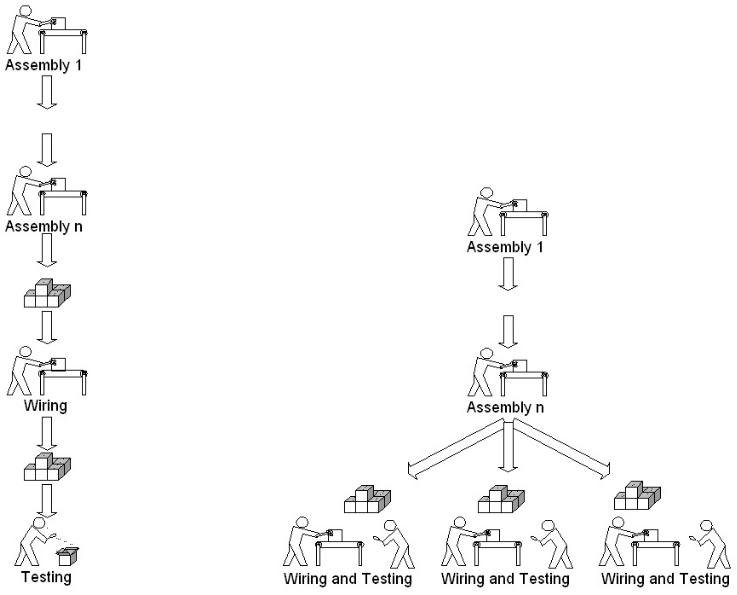
Testing. The final step involves the validation of the completed system. For this, each cabinet is connected to a test station that subjects it to a specially designed set of test signals. If the output of the cabinet is not according to specifications (e.g. due to a cable that is not firmly fixed in place), the test engineer diagnoses the system and corrects the fault.

Of the three stages, assembly is a relatively low skill operation and requires the least amount of training. The wiring step is more complicated, because a great number of connections has to be made between a great number of connectors that all look identical. Moreover, there is a high degree of freedom in the way the connections can be made. Not in terms of the inputs and outputs that have to be connected (these are fixed), but in terms of the order in which the connections are made (i.e. which cables run in front of other cables), the position of the cables (e.g. along the left or the right side of the cabinet) and which cables are tied together. The final step of testing is also complicated, as it involves making diagnoses for the detected anomalies and repairing them. Notice that the high variability in orders exacerbates the complexity in wiring and testing. To address these issues, the management decided to introduce a new setup that allows wirers and test engineers to specialise on product groups. However, this means that production has to be matched to available operator capabilities. The CP-based scheduling system we describe in this paper does exactly that.

The remainder of this paper is organised as follows. In the next section, we describe this change in operations in more detail and show why the use of a scheduling tool is necessary in the new situation. Section 3, then, describes the system that we built in detail. The paper finishes with an evaluation on the system itself and the use of CP in general.

2 Introducing CP

As indicated in the introduction, the wirers and test engineers face a great difficulty because of the huge variety in cabinets. Part of the problem in the original situation, as depicted in Figure 2.a, is that all wirers and all test engineers are considered to be a pool. Assembly puts populated cabinets into a buffer, from which the wirers take their cabinets. In their turn, wirers put finished cabinets into a buffer for testing. In this situation, only a very crude scheduling is employed. Based on the due dates of the orders, it is decided which orders are to be produced on a given day. The assembly stage then starts populating the required cabinets and places them in a buffer for wiring. When a wirer finishes a cabinet, he chooses one of the cabinets from the buffer to work on. Similarly, testers pull a cabinet out of the buffer for testing. In this way, cabinets trickle down the three stages on their way to completion. Aside from the selection of which orders to produce on a given day, no scheduling is performed. This may lead to several issues. For example, when a particular wirer has finished there is a few specific cabinets available in the buffer. Perhaps, the only product groups available are ones that he has no great familiarity with, which may lead to errors. As a second example, consider that a wirer consistently makes the same mistake over a particular period of time. These may be spread over several testers, all of whom will have to diagnose the fault and may not notice the repeated nature of it. Moreover, cabinets from product groups that are hard to wire, may have to wait in the wiring buffer for a long time, since other, easier, cabinets are preferred by the wirers.



(a) Assembly line (b) Cell-based workfloor

Fig. 2. From assembly line to cell-based work floor

To prevent these and similar issues, management had decided to move from what was essentially an assembly line to a (partially) cell-based work floor [6]. In particular, the company wanted to integrate the wiring and testing activities into a *cell*, as depicted in Figure 2.b. Such a cell is made up of a dedicated team consisting of wirers and testers that focuses on a limited number of product groups. One of the key benefits that the company hopes to achieve by this change in setup is an improved quality of wiring and hence lower testing times. However, since the cells are dedicated to certain product groups, the order in which cabinets are assembled becomes much more important. An incorrect sequence may starve certain cells, or lead to building a stock of cabinets waiting to be processed on an already overloaded cell (or indeed both). For this reason, a scheduling tool was required.

The choice of technology was influenced by several factors. Firstly, the problem is a scheduling one in which time needed to be represented at the minute level. Therefore, a CP or IP based approach would be considered. There was uncertainty about what a good schedule would look like and so we would be required to iterate through a number of search strategies and objective functions to achieve the type of schedule they needed. This required an environment to try different options quickly, one of the reasons to work with Ilog OPL Studio. Once we analysed the problem more we could see that here was a variety of complex scheduling constraints which would have been too onerous to model in IP; the high level scheduling concepts of CP-based scheduling provided the ideal medium.

2.1 Related Work

Constraint-based scheduling is a modelling and solving technique successfully used to solve real-world manufacturing problems in, for example, aircraft manufacturing [4], production scheduling [8] and the semiconductor industry [1]. In the area of telecommunications, Little *et al.* [5] have already applied CP to the complex thermal part of the testing of circuit packs. This type of testing is not present in the plant we are concerned with; however, the work was influential in convincing the company of the suitability of the CP technology.

The area of cellular manufacturing is traditionally more about layout and design; where the actual design of the cells, what they are to do, their location and possible performance is of concern. Love [6] gives a clear overview on this type of problem. In particular, Golany *et al.* [2] are concerned with identifying the optimal level of work in progress (WIP) and the best strategy to deal with backlogs, rather than producing a day to day schedule. Also in our case, the cells location, number and they types of operations that they do, has been decided in advance. Ponnabalam *et al.* [9] consider simulation of manufacturing through a cell in light of uncertainty. They propose new heuristics to decide on the allocation of jobs to machines. They are working on theoretical problems here, but also simulating scheduling rather than creating an optimal (or near optimal) schedule.

3 The Program

The basis of the model was developed over the course of a one week site visit. As indicated above, the tool that we used for this was Ilog OPL Studio 3.7.1 with the Ilog Scheduler [7]. Perhaps the most significant reason to use OPL Studio was that we had effectively a time window of one week in which to develop a model which would convince the company of the potential benefits. A rapid prototyping system such as Ilog's OPL 3.7.1 with its variety of solvers seemed a good choice. Additionally, it offers the ability to try a variety of options quickly, while investigating different strategies and objective functions. After the initial model was built and verified, we returned home and finished the details of the model and built a user interface around it. (More on our procedure is outlined in the next section.)

The scheduling system produces a weekly schedule given the list of orders for that week. It does so by iteratively considering each day, extending the schedule built for the previous day(s). For each day, it first produces a schedule for each of the cells, assuring that assembly is possible in principle. Then, in a second phase, it optimises the assembly process. It then fixes this schedule and continues with the next day. Satisfactory results are obtained with optimising each day for 60 CPU-seconds.[2] The user runs the tool twice daily, computing a schedule from scratch each time (but taking into account work in progress). This way, changes to e.g. due dates and availability of material, and disruptions (e.g. machine break-downs) are taken into account. If necessary, the tool can be run when a disruption occurs to immediately take it into account.

Each order o is characterised by a product group $\mathit{group}[o]$, an earliest start time $\mathit{releaseTime}[o]$, a due date $\mathit{dueDate}[o]$, a measure for the availability of components needed for the cabinet $\mathit{matAvailability}[o]$, a priority $\mathit{priority}[o]$ and a partial preference ordering over the shifts $\mathit{preference}[o]$.

3.1 The Model

From Figure 2.b one can see that there are essentially two processes in the new configuration: that of assembly, and the work being carried out on the cells. As the operations within each those processes are a sequence of steps, we opted not to model each of the substeps, but regard each of the cells and the assembly line as a single resource. A number of cabinets is assembled or processed at the same time, however, so we used a *discrete resource* to model these.[3] Unfortunately, the semantics of a discrete resource imply that if we allow n cabinets simultaneously on the assembly line, these might all start at the same time. Clearly this is not the intended behaviour: although a number of cabinets can be assembled simultaneously, these would have to be at different stations (i.e. stages in the process). Therefore, for each discrete resource, we also create a corresponding *unary resource* to regulate the flow.[4] For each cabinet we now create two sets of two of activities: one group to represent the usage of the assembly line, and another to represent the usage of the cell. Both groups consist of an activity for the discrete resource (with a duration equal to the complete process time), and one activity for the unary resource (with a duration chosen according to the desired flow through the resource). Constraints are put in place to ensure that the activities in one group start at the same time. Figure 3 illustrates this behaviour.Here, $\mathit{cellFlowAct}[i]$ is the unary activity on a cell for cabinet i, and $\mathit{wireAndTestAct}[i]$ is the corresponding discrete activity. The duration of the former activity regulates the inflow (it equals the duration of the first wiring step); the duration of the latter equals the amount of time it takes for a cabinet to be wired and tested. Notice that the variability in process times in all but one of the stages is limited for all of the product groups; only the test stage has an uncertain duration. However, the user recognised that this uncertain behaviour

[2] The 60 seconds per day are required to get good results in weeks with a heavy load, where orders have to be shifted to earlier days to achieve the requested due dates. If this is not the case, 20 seconds per day suffice.

[3] Discrete resources "are used to model resources that are available in multiple units, all units being considered equivalent and interchangeable as far as the application is concerned." [3]

[4] A unary resource "is a resource that cannot be shared by two activities." [3]

affects the efficiency of the cell as a whole. Therefore, they implemented a policy to move cabinets that take more than a standard time to test to a special cell. Hence, we are dealing with deterministic processing times, and can use fixed durations in the schedule. In order to implement an efficient search, we make

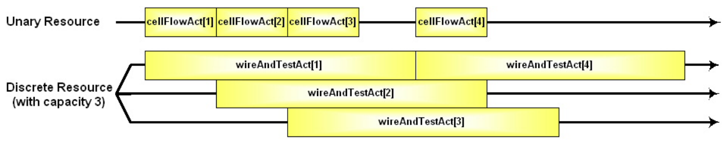


Fig. 3. Example of using both a discrete resource and a unary resource to represent a single entity (in this case: the cell)

use of a boolean matrix *orderOnCellShift*$[o, c, s]$. An entry in this matrix is true iff order o is scheduled on cell c during shift s. Let O be the set of all orders, C the set of all cells, and S the set of all shifts, then the following constraint holds:

$$\forall_{o \in O} \sum_{s \in S} \sum_{c \in C} \mathit{orderOnCellShift}[o, c, s] = 1$$

Orders that are assigned a certain shift are started during that shift (but may only be finished in a later shift):

$$\forall_{o \in O} \sum_{s \in S} \sum_{c \in C} \mathit{orderOnCellShift}[o, c, s] \times \mathit{shift}[s].\mathit{start} \leq \mathit{wireAndTestAct}[o].\mathit{start}$$

$$\forall_{o \in O} \ \mathit{wireAndTestAct}[o].\mathit{start} \leq \sum_{s \in S} \sum_{c \in C} \mathit{orderOnCellShift}[o, c, s] \times \mathit{shift}[s].\mathit{finish}$$

Cells have an associated *availability* matrix which indicates during which shifts they are active, as well as a *capability* matrix that specifies the product groups that these cells can run.

$$\forall_{o \in O} \forall_{c \in C} \sum_{s \in S} \mathit{orderOnCellShift}[o, c, s] \leq \mathit{capability}[c, \mathit{group}[o]]$$

$$\forall_{o \in O} \forall_{c \in C} \forall_{s \in S} \ \mathit{orderOnCellShift}[o, c, s] \leq \mathit{availability}[c, s]$$

Some days, not all orders can be scheduled, and some will have to be postponed to a later date. A boolean variable *delayed*$[o]$ is used to indicate that the due date may be missed (but the order is still included in today's schedule). Another variable *postponed*$[o]$ (which implies *delayed*$[o]$) indicates that the order will be left for the next scheduling iteration.

$$\forall_{o \in O} \ (1 - \mathit{delayed}[o]) \times \mathit{wireAndTestAct}[o].\mathit{end} \leq \mathit{dueDate}[o]$$

In addition to the above constraints, there are constraints linking the activities to their corresponding resources, precedence constraints as well as special constraints that keep track of the assignments that have been made so far, in order to make decisions during search. The criterion is a minimisation of the makespan, combined with the number of orders that have to be postponed. However, as we shall see next, the search is geared towards clustering of product groups to ensure few changeovers during a shift.

3.2 Search

A custom search procedure is used to produce the schedules for the cells, see Algorithm 1. It considers each order in turn, ordered by their priorities, the availability of components and their release times. It first tries to add the order to today's schedule, satisfying the due date constraints. To do so, it tries to assign the order to each valid shift / cell combination. It does so following an order. In this case, it first tries shifts with the highest preferences, and cells that have already one or more cabinets of the same product group assigned. The latter is to direct towards solutions that have a low number of different product groups (and hence changeovers) that are assigned to a cell. If a shift / cell combination is chosen, the order is assigned the earliest possible start time on that cell during the shift.

If all these options fail, the search procedure next considers running late with the order. This means that the order is started *before* it's due time, but is only finished *after*. Notice that we only have to consider the last possible shift of the current time window in this case. Starting the processing of a cabinet during one of the earlier shifts will result in the cabinet being finished in time, and the previous step of the search procedure has shown that this cannot lead to a valid solution. Finally, if neither of the previous steps result in a valid assignment for the order, it is left for the next day's schedule.

Once the cells have been assigned, a second search procedure is started that optimises the assembly operations. The first stage finds ensures the existence of a valid schedule for the assembly operations (to ensure that the schedule for the cells is valid), but it does not try to optimise it, as the focus is on optimising the usage of the cells. In this second phase, the schedule for the cells is kept fixed, while the maximum time between the completion of assembly of a cabinet and the start of the cell operations on that cabinet is minimised. The search procedure for this stage is shown in Algorithm 2. This greedy algorithm iteratively selects the order for which the minimum possible time that is spent in the buffer between assembly and wiring is currently the largest.[5] For this order, it tries to assign the best possible time left, or removes it from the domain. It does so for all orders, until all orders have been assigned a time to start the assembly.

[5] The **dmax**(v) function returns the maximum value in the domain of the variable v. The function **unbound**(v) returns *true* if the variable v is unbound (i.e. has not been assigned a value yet), and *false* otherwise.

Algorithm 1. Search procedure for the first stage

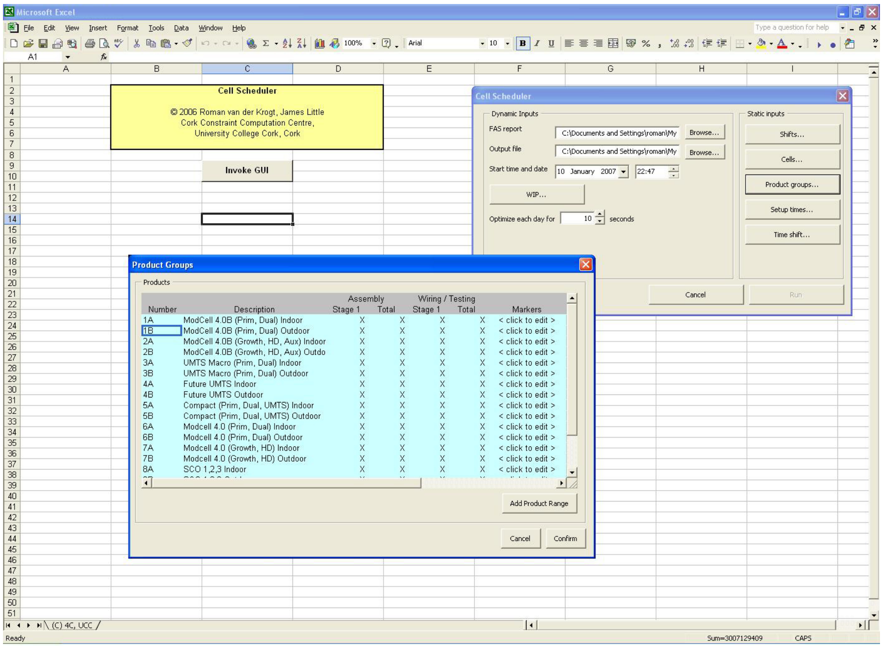
```
begin
    forall orders $o \in O$ ordered by increasing $\langle priority[o],\ matAvailability[o],$
    $releaseTime[o]\rangle$
        try
            // first try to schedule it for today
            $delayed[o] = 0$
            $postponed[o] = 0$
            forall shifts $s \in S$ ordered by increasing $\langle preference[o], s\rangle$
                forall cells $c \in C$ ordered by increasing $\langle usage, s\rangle$
                    try
                        $orderOnCellShift[o, c, s] = 1$
                        assign minimum time
                    or
                        $orderOnCellShift[o, c, s] = 0$
        or
            // consider finishing too late
            $delayed[o] = 1$
            $postponed[o] = 0$
            select $s \in S$ : $s$ is the last shift in the current window
            forall cells $c \in C$ ordered by increasing $\langle usage, s\rangle$
                try
                    $orderOnCellShift[o, c, s] = 1$
                    assign minimum time
                or
                    $orderOnCellShift[o, c, s] = 0$
        or
            // wait for tomorrow's schedule
            $delayed[o] = 1$
            $postponed[o] = 1$
end
```

Algorithm 2. Search procedure for the second stage

```
begin
    while $\exists o \in O$ : unbound($assemblyStart[o]$) do
        select $o \in O$ : unbound($assemblyStart[o]$) ordered by decreasing
        $\langle cellStart[o] - \mathbf{dmax}(assemblyStart[o])\ \rangle$
        try
            $assemblyStart[o] = \mathbf{dmax}(assemblyStart[o])$
        or
            $assemblyStart[o] < \mathbf{dmax}(assemblyStart[o])$
end
```

3.3 The GUI

The user interface is built in Microsoft Excel using Visual Basic. The reason for this is twofold. On the one hand, the users are comfortable using the Microsoft Office suite, which means that no extra training is required. On the other hand, it allowed for easy integration with the existing Work floor Management System that tracks all activities and can generate reports in the Excel file format.



Cell Scheduler

© 2006 Roman van der Krogt, James Little
Cork Constraint Computation Centre,
University College Cork, Cork

Cell Scheduler

Dynamic Inputs

FAS report C:\Documents and Settings\roman\My

Output file C:\Documents and Settings\roman\My

Start time and date 10 January 2007 22:47

Optimize each day for 10 seconds

Static inputs

Product Groups

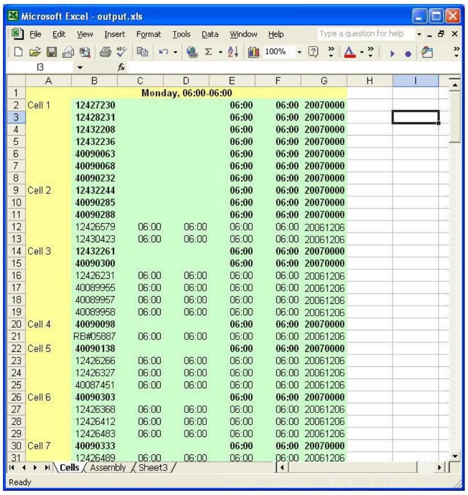
Products

		Assembly		Wiring / Testing		
Number	Description	Stage 1	Total	Stage 1	Total	Markers
1A	ModCell 4.0B (Prim, Dual) Indoor	X	X	X	X	< click to edit >
1B	ModCell 4.0B (Prim, Dual) Outdoor	X	X	X	X	< click to edit >
2A	ModCell 4.0B (Growth, HD, Aux) Indoor	X	X	X	X	< click to edit >
2B	ModCell 4.0B (Growth, HD, Aux) Outdo	X	X	X	X	< click to edit >
3A	UMTS Macro (Prim, Dual) Indoor	X	X	X	X	< click to edit >
3B	UMTS Macro (Prim, Dual) Outdoor	X	X	X	X	< click to edit >
4A	Future UMTS Indoor	X	X	X	X	< click to edit >
4B	Future UMTS Outdoor	X	X	X	X	< click to edit >
5A	Compact (Prim, Dual, UMTS) Indoor	X	X	X	X	< click to edit >
5B	Compact (Prim, Dual, UMTS) Outdoor	X	X	X	X	< click to edit >
6A	Modcell 4.0 (Prim, Dual) Indoor	X	X	X	X	< click to edit >
6B	Modcell 4.0 (Prim, Dual) Outdoor	X	X	X	X	< click to edit >
7A	Modcell 4.0 (Growth, HD) Indoor	X	X	X	X	< click to edit >
7B	Modcell 4.0 (Growth, HD) Outdoor	X	X	X	X	< click to edit >
8A	SCO 1,2,3 Indoor	X	X	X	X	< click to edit >

Fig. 4. Screenshot of the GUI

Figure 4 shows the user interface. The window in the top right (labeled "Cell Scheduler") forms the heart of the interface. On the left-hand side, it allows the input of some general data (the input file for orders and the output file for the resulting schedule, for example) and on the right-hand side there is a number of buttons that allow the user to specify the more static input to the scheduler. This includes things like the number of cells, their capabilities and availabilities, changeover times between product groups, and the product groups themselves. (This last window is shown in the foreground.)

The output of the scheduling tool is presented as an Excel file, see Figure 5. This file consists of two sheets: one for assembly and one for the cells. On each tab, it shows per shift which work is to be done and when and where each activity is to take place. The rows in bold face correspond to work that was in progress when the scheduling tool was run. Each row lists the order number, the start en finish times of assembly (if applicable), the start and finish times of wiring and testing(i.e. the


Microsoft Excel - output.xls
File Edit View Insert Format Tools Data Window Help
Type a question for help
100%
I3
Cells
Assembly
Sheet3
Ready

	A	B	C	D	E	F	G	H	I
1				Monday, 06:00-06:00					
2	Cell 1	**12427230**			**06:00**	**06:00**	**20070000**		
3		**12428231**			**06:00**	**06:00**	**20070000**		
4		**12432208**			**06:00**	**06:00**	**20070000**		
5		**12432236**			**06:00**	**06:00**	**20070000**		
6		**40090063**			**06:00**	**06:00**	**20070000**		
7		**40090068**			**06:00**	**06:00**	**20070000**		
8		**40090232**			**06:00**	**06:00**	**20070000**		
9	Cell 2	**12432244**			**06:00**	**06:00**	**20070000**		
10		**40090285**			**06:00**	**06:00**	**20070000**		
11		**40090288**			**06:00**	**06:00**	**20070000**		
12		12426579	06:00	06:00	06:00	06:00	20061206		
13		12430423	06:00	06:00	06:00	06:00	20061206		
14	Cell 3	**12432261**			**06:00**	**06:00**	**20070000**		
15		**40090300**			**06:00**	**06:00**	**20070000**		
16		12426231	06:00	06:00	06:00	06:00	20061206		
17		40089955	06:00	06:00	06:00	06:00	20061206		
18		40089957	06:00	06:00	06:00	06:00	20061206		
19		40089958	06:00	06:00	06:00	06:00	20061206		
20	Cell 4	**40090098**			**06:00**	**06:00**	**20070000**		
21		RB#05887	06:00	06:00	06:00	06:00	20061206		
22	Cell 5	**40090138**			**06:00**	**06:00**	**20070000**		
23		12426266	06:00	06:00	06:00	06:00	20061206		
24		12426327	06:00	06:00	06:00	06:00	20061206		
25		40087451	06:00	06:00	06:00	06:00	20061206		
26	Cell 6	**40090303**			**06:00**	**06:00**	**20070000**		
27		12426368	06:00	06:00	06:00	06:00	20061206		
28		12426412	06:00	06:00	06:00	06:00	20061206		
29		12426483	06:00	06:00	06:00	06:00	20061206		
30	Cell 7	**40090333**			**06:00**	**06:00**	**20070000**		
31		12426489	06:00	06:00	06:00	06:00	20061206		

Fig. 5. Screenshot of the resulting output (censored)

activity on the cell) and the original due date. The cell sheet (which is shown in Figure 5) groups the orders by cell, whereas the assembly sheet merely shows the orders over time.

4 Evaluation

4.1 From Prototype to Production

The scheduling system is currently in the final stages of testing. As said before, the project started with the authors visiting the site for a week in October 2006. During the first two days of the week, we had meetings with the key figures in the factory and were painted a detailed picture of the processes going on. We then worked on a prototype model, confirming our decisions and asking for clarifications as we needed them. Although very intense (for both sides), this proved to be a very fruitful setup: by the end of the week we could present an initial version of our system. After presenting our first results, the company was confident that the CP-based scheduling approach that we proposed could deliver the tool they needed. So confident, in fact, that they decided to buy the necessary licenses, a considerable investment, during the course of our presentation. (The option of re-engineering the model in another language was also considered, but rejected because of time constraints.) We took the model home, further refined it, and added the GUI as discussed in Section 3.3. Early January, we produced the first schedules in parallel to the schedules the company made themselves. Eventually, when enough confidence in the schedules was gained, the system's schedules were used as the basis for the company's own schedules. From the second half of March, the company is using the schedules as is. As all recent issues are of a cosmetic nature, we expect to move into full production mode shortly. This would mean that the project has taken roughly eight months from the start. Note, however, that due to issues obtaining a license to OPL for the company the project was stalled during December, and it was worked on part-time. Taking this into account, the project has taken three to four months of full-time work.

It is hard to quantify the direct benefit that the scheduling tool has delivered, as it is part of the overall implementation of a different way of working. Moreover, the transition to the new setup has only recently been completed. However, already, the user has observed significant improvements in manufacturing interval, work in process inventory, first test yield, head count, quality and delivery performance. Although these benefits are mainly achieved because of the change to a cell layout, the scheduling tool is crucial in realising the full potential of this layout. The feeling is that the return on investment is considerable, although no figures have been calculated yet. Due to its success, the company considers extending the cell-based work floor concept to other areas of the factory. A CP-based scheduler would be part of that extension.

4.2 Lessons Learned

The week that we spent on-site developing the first model was intensive (for all parties), but great to kick start the project. By the end of the week, the outline of the complete system could be seen, which acted as a motivator to keep things going. However, to achieve this required commitment from the people in the factory, who had to spend a lot of time interacting with us. The engineers at the site had experience with the Theory of Constraints. As such, they had intuitive notions of what a constraint is and how to identify them. This helped us greatly in the initial modelling and getting the feedback on that model. Both these factors were required for the success of our visit. If these requirements are met, however, we can recommend this approach as it helps the developers to get to grips with the problem quickly, and builds a strong relationship between the developer and the user.

From our point of view, the main lesson learned is that it is harder than it seems to successfully deploy a system. This may seem obvious to some, but the amount of work it takes to go from prototype to a system robust enough to withstand user interaction is considerable. Moreover, the variety of situations that is encountered in practice means that the model itself has to be robust enough to allow for very different input profiles. This either calls for a large data set during development, or a period of building this robustness during the testing process. As there was no data available on the new layout when we started, we opted for the latter. In this case it is important to make the user realise that the initial phase of testing will be slow and not not very smooth at times, as the tool is adjusted to match the reality on the shop floor.

4.3 Final Remarks

This project shows the strength of the combination of lean manufacturing techniques and (CP-based) scheduling. The former techniques allow a business to take a critical look at its operations to identify areas for improvement, whereas the latter technique can be used to realise the potential of the improvements. This becomes more and more an issue as businesses move to more advanced methods of operation. As the user puts it: "*The amount of constraints the program has to handle points out the need for it. Scheduling manually would not allow us to service our customers as well.*"

Acknowledgements

Roman van der Krogt is supported by an Irish Research Council for Science, Engineering and Technology (IRCSET) Postdoctoral Fellowship. James Little is supported by Science Foundation Ireland under Grant- 03/CE3/I405 as part of the Centre for Telecommunications Value-Chain-Driven Research and Grant-05/IN/I886.

The authors would like to thank everybody at the Systems Integration Center in Columbus, Ohio for their warm welcome and support.

References

1. Bixby, R., Burda, R., Miller, D.: Short-interval detailed production scheduling in 300mm semiconductor manufacturing using mixed integer and constraint programming. In: The 17th Annual SEMI/IEEE Advanced Semiconductor Manufacturing Conference (ASMC-2006), pp. 148–154. IEEE Computer Society Press, Los Alamitos (2006)
2. Golany, B., Dar-El, E.M., Zeev, N.: Controlling shop floor operations in a multi-family, multi-cell manufacturing environment through constant work-in-process. IIE Transactions 31, 771–781 (1999)
3. Ilog: Ilog OPL Studio 3.7.1 help files (2005)
4. Bellone, J., Chamard, A., Fischler, A.: Constraint logic programming decision support systems for planning and scheduling aircraft manufacturing at dassault aviation. In: Proceedings of the Third International Conference on the Practical Applications of Prolog, pp. 111–113 (1995)
5. Creed, P., Berry, S., Little, J., Goyal, S., Cokely, D.: Thermal test scheduling using constraint programming. In: Proceedings of the 12th IFAC Symposium on Information Control Problems in Manufacturing (2006)
6. Love, D.: International Encyclopedia of Business and Management. In: The Design of Manufacturing Systems, vol. 4, pp. 3154–3174. Thompson Business Press (1996)
7. Nuijten, W., Le Pape, C.: Constraint-based job shop scheduling with IILOG SCHEDULER. Journal of Heuristics 3, 271–286 (1998)
8. Le Pape, C.: An application of constraint programming to a specific production scheduling problem. Belgian Journal of Operations Research, Statistics and Computer Science (1995)
9. Ponnambalam, S.G., Aravindan, P., Reddy, K.R.R.: Analysis of group scheduling heuristics in a manufacturing cell. The International Journal of Advanced Manufacturing Technology 15, 914–932 (1999)