

Production Scheduling With a Spreadsheet

- Tony Rice,
production-scheduling.com,
South Africa

Introduction

For years spreadsheets have been used to design and prototype scheduling systems. They have now grown up, and are being used to develop serious production scheduling applications. A comprehensive tutorial available at no charge at www.production-scheduling.com and this paper summarizes it, introduces some of the concepts, and illustrates what is possible with a spreadsheet. It is aimed at spreadsheet literate people who are involved in planning and scheduling production activities. The techniques and formulas are being used by manufacturing companies daily; this is a practical, not an academic, exercise.

Background

I have designed, built and implemented production scheduling systems for manufacturing companies for over 11 years. When PC's and spreadsheets were less capable than they are now, we used spreadsheets to design and prototype scheduling algorithms, and to train on some of the principles of scheduling. Prototype designs were then handed over to software developers to write in more resilient and efficient programming languages.

Impatient clients pressured us to throw several thousand records of data at the prototypes and use them for live scheduling, before handing them to the software developers. So, in order to 'shoehorn' a big scheduling task into a small PC, we recorded macros that wrote a formula, copied it down, overwrote the cells with values, then moved on to the next column, so that no memory consuming live formulas were left behind. Typically, most of the macro code prepared downloaded data for scheduling, and generated reports from the schedule, with only a small portion of the macro calculating the schedule itself. We ended up with big cumbersome macro driven scheduling systems that locks out ordinary, spreadsheet literate people.

Thankfully we now have Pentium III's which will handle large amounts of data, and have features such as Excel's PivotTable which will re-arrange and summarize data for scheduling, and prepare reports **without** writing macros. It makes the job, of building a scheduling system with a spreadsheet, a whole lot easier, and within the capability of the average spreadsheet user.

The Case for Spreadsheets

There is a worldwide shortage of IT and programming skills, but in most businesses a good depth of spreadsheet literacy has developed. With some assistance, manufacturers can harness these skills, uplift them, and end up with user- maintainable production scheduling systems. The cost is considerably less than implementing one of the scheduling packages, and the risk of being stranded without IT support, is lower. It must be recognized, however that spreadsheets may be good for manipulating data, but they are not transaction processors, and it is wise to interface to a host ERP system.

Structured and Disciplined Approach

Spreadsheets have earned themselves a bad reputation amongst software purists, because they can, and often are, used in an unstructured way. Building a scheduling system requires a structured and disciplined approach. Please resist the trap, that many fall into, of creating a table on a single worksheet that looks like the report that you want to see. The approach used here is to create lists in the form of databases, with a heading at the top of

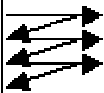
each column, and with universal formulas that can be copied and pasted down a column, and work on every row. If all the calculations are done in a structured database, then reports, with sub-totals and charts, can easily be created with a PivotTable.

Time Cascades Downwards

Jobs	Hours	Start	Stop
Job A	7	0	7
Job B	12	7	19
Job C	4	19	23
Job D	5	23	28
Job E	8	28	36

Here is a very simple finite schedule. It starts at hour zero, and the next job starts when the previous one finishes.

Time cascades downwards in a pattern like this:



Re-sequencing the Schedule

Seq	Jobs	Hours	Start	Stop	Due	On time
1	Job A	7	0	7	24	TRUE
2	Job B	12	7	19	24	TRUE
3	Job C	4	19	23	24	TRUE
4	Job D	5	23	28	24	FALSE
5	Job E	8	28	36	24	FALSE

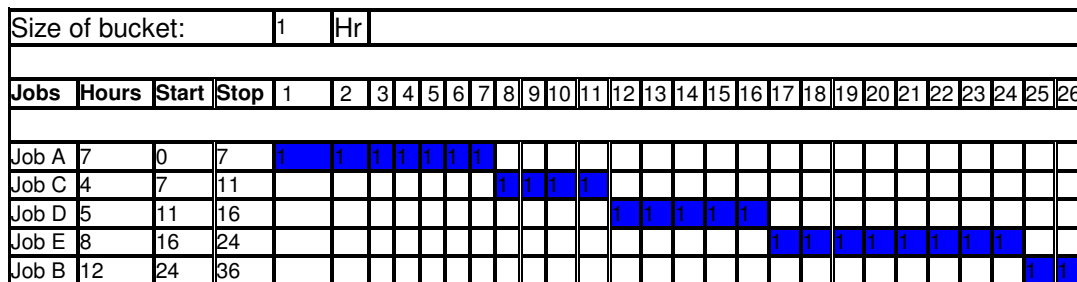
Here is the same schedule with a job sequence and a due date added. You can see that there is 36 hours of work, which is all due in 24 hours time - we have a problem. This is a good opportunity to illustrate the essential difference between finite scheduling and capacity planning. Capacity planning would tell you that you have a problem by saying you are 150% loaded. Finite scheduling tells you about the problem by saying that out of 5 jobs, two are going to be late. Lets see what it would look like if we left Job B till last. We change sequence 2 to 6 and hit the sort icon.

Seq	Jobs	Hours	Start	Stop	Due	On time
1	Job A	7	0	7	24	TRUE
3	Job C	4	7	11	24	TRUE
4	Job D	5	11	16	24	TRUE
5	Job E	8	16	24	24	TRUE
6	Job B	12	24	36	24	FALSE

Now only one job is going to be late. I would argue that finite scheduling gives you better management information than capacity planning does. In this simplistic example it is all very obvious, but if you had 500 jobs to do in the next 6 weeks, you would need a tool to do the calculations for you.

A Simple Gantt Chart

The mistake that many people make when constructing a spreadsheet based schedule, is to start by making a Gantt chart with time running from left to right. The technique here is to recognize that a Gantt chart is merely a report that represents time in buckets, whereas the calculations run downwards and are bucketless.



The formula in each cell of the Gantt chart looks up at the top row, and flags the cell if it lies between the stop and start of the job, and conditional formatting is used to color it. The size of the bucket is variable, for example:

Size of bucket:	2	hrs																																																							
Jobs	Hours	Start	Stop	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40	42																																	
Job A	7	0	7	1	1	1																																																			
Job C	4	7	11				1	1																																																	
Job D	5	11	16							1	1	1																																													
Job E	8	16	24													1	1	1	1																																						
Job B	12	24	36																									1	1	1	1	1	1																								

Change-over Matrix

Often change-over time can be saved by undertaking jobs in a certain sequence. A typical example would be a paint line where a change from a dark color to a light color requires a much longer cleaning process than from a light color to a dark color. The differential change over times are expressed in the matrix to the left, and the schedule on the right has an extra column for change-overs (C/O).

	Hours												
To/From	Job A	Job B	Job C	Job D	Job E	Seq	Jobs	Hours	C/O	Start	Stop	Due	On Time
Job A	0.0	0.5	1.0	1.5	2.0	1	Job B	11	0	0	11	24	TRUE
Job B	0.1	0.0	0.5	1.0	1.5	2	Job E	7	1.5	11	19.5	24	TRUE
Job C	0.2	0.1	0.0	0.5	1.0	3	Job A	6	0.4	19.5	25.9	24	FALSE
Job D	0.3	0.2	0.1	0.0	0.5	4	Job D	4	1.5	25.9	31.4	24	FALSE
Job E	0.4	0.3	0.2	0.1	0.0	5	Job C	3	0.1	31.4	34.5	24	FALSE
									Total Change-overs	3.5	Hrs		

It is assumed that the first job has already been set up, so has a change-over time of zero. Now re-sequencing the schedule becomes a little more challenging. The above sequence would mean that 3 jobs would be late and 3.5 hours would be spent on change-overs, so let us see if we can sequence more intelligently:

Seq	Jobs	Hours	C/O	Start	Stop	Due	On time
1	Job E	7	0	0	7	24	TRUE
2	Job D	4	0.1	7	11.1	24	TRUE
3	Job C	3	0.1	11.1	14.2	24	TRUE
4	Job A	6	0.2	14.2	20.4	24	TRUE
5	Job B	11	0.5	20.4	31.9	24	TRUE
Total Change-overs			0.9	Hrs			

Again, we change the sequence numbers and click in the sort icon. That's better, only one late job and 0.9 of an hour on change-overs. By reading the change-over matrix the behavior of the schedule is much more sensitive to sequence changes.

Base 1900 Date

NOW: 36982.9016635417 Days since 1 January 1900	
Here are some different formats of the date:	
21:38:24	Sun
09:38:24 PM	Sunday
09:38 PM	Apr
4/1	April
4/1/01	2001
1-Apr	01-Apr-01
April-01	April 1, 2001

Up to now we have referred to points in time as, for example, "36 hours into the schedule" from hour zero. Excel and other spreadsheets use days, rather than hours as the primary measure of time, and the starting point is midnight on 1st of January of the year 1900. All time calculations are done in days, but there are enough decimal places of a day to measure less than a 3000th of a second. This makes a spreadsheet particularly useful for date and time calculations, as we only have to concern ourselves with one unit of measure, a day. A date or elapsed time measured in days may be formatted and viewed as seconds, minutes, hours, day of the week, day of the month, weeks, months or years.

If we express our schedule using base 1900 dates our schedule would now look like this, and the starting point will no longer be hour zero, so we have to establish a starting point for the schedule. In this case midnight on 1 April this year.

Start Schedule:		of		01 Apr 12:00 AM	
Jobs	Hours	Days	Start	Stop	
Job A	7	0.2917	01 Apr 12:00 AM	01 Apr 07:00 AM	
Job B	12	0.5000	01 Apr 07:00 AM	01 Apr 07:00 PM	
Job C	4	0.1667	01 Apr 07:00 PM	01 Apr 11:00 PM	
Job D	5	0.2083	01 Apr 11:00 AM	02 Apr 04:00 PM	
Job E	8	0.3333	02 Apr 04:00 AM	02 Apr 12:00 PM	

Working With a Calendar

Our simple schedule assumes that we work 24 hours a day for 7 days per week, which is fine if we are scheduling a continuous process such as an oil refinery or a paper mill, but most manufacturers work to a shift calendar. Here is an example of a shift calendar:

Hrs	Day	Begin	End	Cum. Days			
2	Mon	04 Jun 08:00 AM	04 Jun 10:00 AM	0.083	Duration:	22.8	Hrs
2.75	Mon	04 Jun 10:15 AM	04 Jun 01:00 PM	0.198	Start:	04 Jun 12:27 PM	Mon
2	Mon	04 Jun 01:30 PM	04 Jun 03:30 PM	0.281			
2.25	Mon	04 Jun 03:45 PM	04 Jun 06:00 PM	0.375	Start row:	7	Calc 1
3	Mon	04 Jun 07:00 PM	04 Jun 10:00 PM	0.500	Cum. day:	0.175	Calc 2
2	Tue	05 Jun 08:00 AM	05 Jun 10:00 AM	0.583	Stop row:	17	Calc 3
2.75	Tue	05 Jun 10:15 AM	05 Jun 01:00 PM	0.698	Stop:	06 Jun 11:15 AM	Wed
2	Tue	05 Jun 01:30 PM	05 Jun 03:30 PM	0.781			
2.25	Tue	05 Jun 03:45 PM	05 Jun 06:00 PM	0.875			
3	Tue	05 Jun 07:00 PM	05 Jun 10:00 PM	1.000			
2	Wed	06 Jun 08:00 AM	06 Jun 10:00 AM	1.083			
2.75	Wed	06 Jun 10:15 AM	06 Jun 01:00 PM	1.198			
2	Wed	06 Jun 01:30 PM	06 Jun 03:30 PM	1.281			

2.25	Wed	06 Jun 03:45 PM	06 Jun 06:00 PM	1.375	
3	Wed	06 Jun 07:00 PM	06 Jun 10:00 PM	1.500	

Each row of the calendar is a working period, and the "Cum. Days" column tracks the cumulative days from the beginning of the calendar. The calculations to the right of the calendar assume that we start a job at 12:27pm on 4 June and work for 22.8 hours, when will the job stop? There are a series of formulas that reference the calendar and calculate the stop time, and the detailed derivation of the formulas is set out in the tutorial. Broadly the formulas work in this way: The job starts during the working period on row 7, 0.175 days from the start of the calendar, and the job will stop during row 17 of the calendar at 11:15am on 6 June.

This is an example of how the schedule looks when we work through the calendar:

Start of Schedule		20 JUN 10:30 AM					
Jobs	Hours	Days	Start	Calc1	Calc2	Calc3	Stop
Job A	7	0.292	20 Jun 10:30 AM	65	5.844	68	20 Jun 07:15 PM
Job B	12	0.500	20 Jun 07:15 PM	68	6.135	73	21 Jun 07:15 PM
Job C	4	0.167	21 Jun 07:15 PM	73	6.635	74	22 Jun 09:15 AM
Job D	5	0.208	22 Jun 09:15 AM	74	6.802	76	22 Jun 03:00 PM
Job E	8	0.333	22 Jun 03:00 PM	76	7.01	80	25 Jun 02:00 PM

Jobs That Pass Through Multiple Work Centers

So far the start of a job is determined by the stop of the previous job, but where a job goes through several operations on different work stations, a work station may have to wait for the job to emerge from the previous work station before it can start. Here is an example of Jobs A, B and C passing through work centers 6, 7 and 8:

Start of Schedule:				4/6 02:00													
						4/6		5/6			6/6						
W/C	Jobs	Job/Op	Duration Hrs	Prev. Op.	Stop of Prev. Op.	Wait Hrs	Start	Stop	00:00	08:00	16:00	00:00	08:00	16:00	00:00	08:00	16:00
6	Job A	A/1	12	A/0		0	4/6 02:00	4/6 14:00	6	6							
6	Job B	B/1	18	B/0		0	4/6 14:00	5/6 08:00	2	8	8						
6	Job C	C/1	15	C/0		0	5/6 08:00	5/6 23:00				8	7				
7	Job A	A/2	10	A/1	4/6 14:00	12	4/6 14:00	5/6 00:00	2	8							
7	Job B	B/2	14	B/1	5/6 08:00	8	5/6 08:00	5/6 22:00				8	6				
7	Job C	C/2	6	C/1	5/6 23:00	1	5/6 23:00	6/6 05:00						1	5		
8	Job A	A/3	8	A/2	5/6 00:00	22	5/6 00:00	5/6 08:00				8					
8	Job B	B/3	11	B/2	5/6 22:00	14	5/6 22:00	6/6 09:00					2	8	1		
8	Job C	C/3	14	C/2	6/6 05:00	0	6/6 09:00	6/6 23:00							7	7	

Each operation looks up the stop time of the previous operation (if there is one), it also looks at the stop of the previous job on the work center, and uses the later of the two to establish the start time. Note the gaps where a work center has to wait.

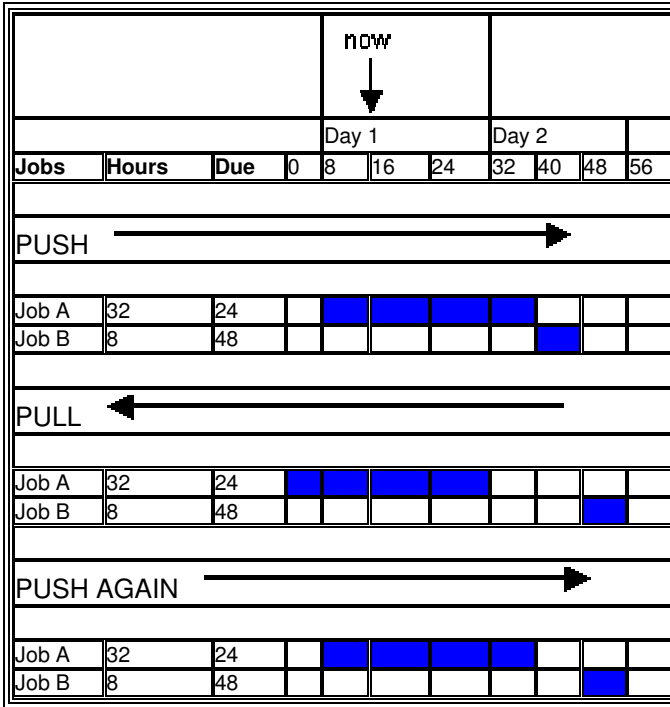
A similar technique is used when two or more manufactured components meet at an assembly operation. A separate column is set up for each component, and the stop dates determine the start of the assembly operation.

Notice also that the Gantt chart buckets are 8 hour periods labeled with a base 1900 date and time, and that instead of simple flags, the buckets show the number of job hours in each bucket. The Gantt chart formula is more involved than a simple flag, and a full explanation is in the tutorial.

Now is a good point to introduce some Theory of Constraints (TOC) thinking. Have a look at Job C going through work center 7. It starts at 11:00pm as soon as it comes off work center 6, runs till 5:00am, then waits for 4 hours for work center 8 to become available. Why start it earlier than you have to, and have it lying in WIP?

Push-Pull-Push 3 Pass Logic

In Eli Goldratt's book, *The Haystack Syndrome*, he describes a scheduling technique which he likens to "leveling the ruins" with a bulldozer in 3 passes, forwards, backwards, and forwards again. A very similar 3 pass algorithm can be done with a spreadsheet. The details of the calculations are set out in the tutorial, but in essence this is how it works. There are two jobs on the same work center, one due in 24 hours time and the other in 48 hours:



The first pass starts now and does both jobs as soon as possible

The second pass starts each job just in time to meet the due date. Job A will have to have started yesterday. Obviously, this is not possible.

The third pass pushes Job A to a feasible start time, and leaves Job B where it was, to leave a schedule which is the best due date performance that can be achieved, but with a "pull" approach.

When the algorithm is applied to a multiple work center schedule, it has the effect of delaying an operation until the next operation is ready for it, discouraging the building of WIP for the sake of keeping machines busy. The second pass works backwards, with the calculations going up the columns instead of down, and each operation may be influenced by the start of the next operation.

Repetitive Production and Transfer Batches

Up to now our examples have been of jobs with a duration in hours. Here is an example of repetitive production, each job being for batches of multiple products that run through each work center at a rate per hour. First let us assume that the entire batch must be complete at a work center before it can go to the next work center.

Start of Schedule:					14/3 04:00											
Transfer batch:																
					14/3			15/3			16/3			17/3		
Work Center	Product	Qty	Units per Hour	Wait Hours	00:00	08:00	16:00	00:00	08:00	16:00	00:00	08:00	16:00	00:00	08:00	16:00
6	Prod A	180	13	0.0	52	104	24									
6	Prod B	300	12	0.0			74	96	96	34						
6	Prod C	220	15	0.0					77	120	23					
7	Prod A	180	21	13.8			129	51								
7	Prod B	300	19	16.4					98	152	50					
7	Prod C	220	22	0.0							118	102				
8	Prod A	180	16	22.4				89	91							
8	Prod B	300	17	21.0							91	136	73			
8	Prod C	220	18	0.0										67	144	9

Now see the impact of transferring products between work centers in batches of 10. What took 4 days, now takes less than 3 days:

Start of Schedule:					14/3 04:00											
Transfer batch:																
					14/3			15/3			16/3			17/3		
Work Center	Product	Qty	Units per Hour	Wait Hours	00:00	08:00	16:00	00:00	08:00	16:00	00:00	08:00	16:00	00:00	08:00	16:00
6	Prod A	180	13	0.0	52	104	24									
6	Prod B	300	12	0.0			74	96	96	34						
6	Prod C	220	15	0.0						77	120	23				
7	Prod A	180	21	0.5	46	104	30									
7	Prod B	300	19	0.1			68	96	96	40						
7	Prod C	220	22	0.0						70	121	30				
8	Prod A	180	16	1.1	38	104	38									
8	Prod B	300	17	0.0			60	96	96	48						
8	Prod C	220	18	0.0						61	121	38				

If a fast operation follows a slow operation, as in this example, it will stop and start as it waits for each transfer batch to reach it. The 3 pass algorithm avoids this by delaying the start of the fast operation so that it catches up with the slow operation by the time the job finishes. To ensure that the second operation does not finish before the first, it needs to look up the stop, as well as the start of the previous operation. For simplicity these detailed calculations are hidden in this example, but they are explained in the tutorial.

Notice that the Gantt chart now shows the number of units produced in each time bucket. This is done by taking the hours per bucket, as in the previous Gantt chart, and multiplying by a rate per hour.

Sequencing a Multiple Work Center Schedule

These simple examples show each product with the same routing, work center 6 then 7 then 8. Some processes work this way, so sequencing may simply be a matter of sorting operations by due date within work center. Manufacturing processes where the products follow very different routings, may require more sophisticated sequencing rules to be calculated. One such calculation works as follows:

- the loading on each work center, in hours, is calculated
- for each job the most heavily loaded work center that it passes through, is identified
- we consider this to be the job's constraint
- assuming infinite capacity (for now) we calculate the latest time that the job must start on the constraint, in order to meet its due date
- we then sort by this date to determine the sequence through each work center

Often it is a good idea to allow the sequences to be overridden manually to see how the schedule responds to changes in sequence.

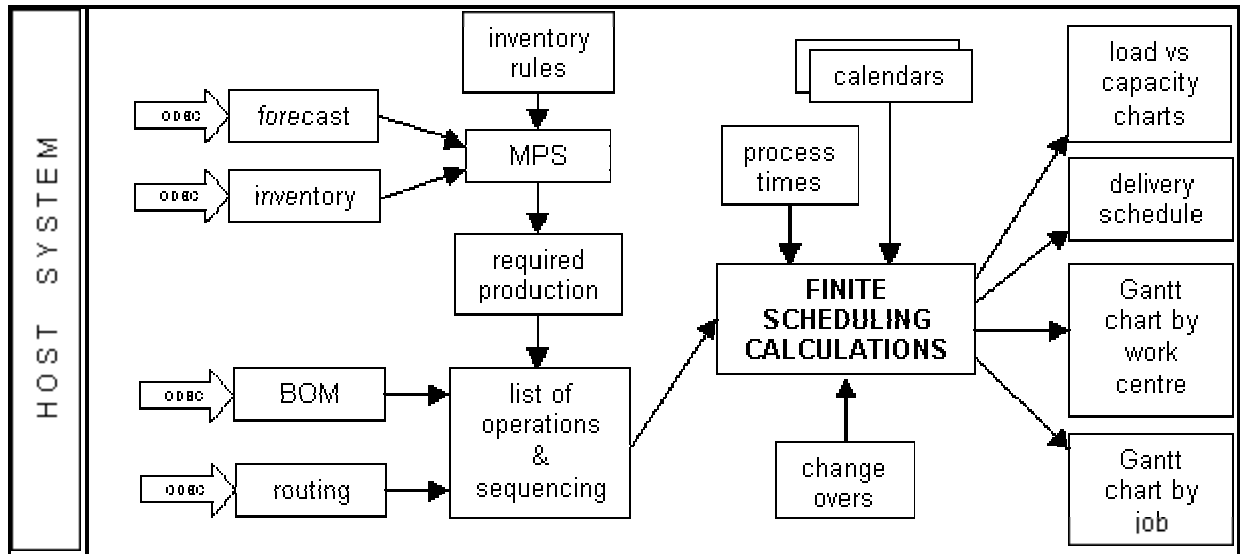
Putting it All Together

In my experience of several hundred scheduling systems, each one is different, but a typical spreadsheet scheduling system may have the following features:

- ODBC interfaces with a host ERP or MRP system
- routings through multiple work centers
- a BOM with several manufactured components at an assembly operation
- a separate calendar for each work center
- a change-over matrix for each work center
- repetitive production
- different transfer batch sizes for each product between each work center

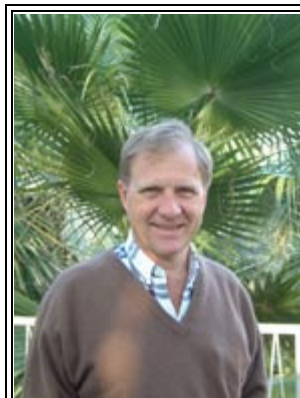
- demand that may explode into several thousand operations
- a rough pre-schedule to establish a sequence for each operation
- the push-pull-push 3 pass algorithm
- Gantt charts with variable sized time buckets

The flow of data through the system would typically be as follows:



Such a system would:

- have about 20 pages in a workbook
- be up to 30Mb in size
- have finite scheduling calculations with up to 150 columns and several thousand rows
- contain about 10 PivotTables
- have NO macro code
- take from 5 to 30 seconds to re-schedule



Author:
Tony Rice
production-scheduling.com

24 Kenilworth Drive,
Kloof 3610, KwaZulu/Natal,
South Africa

Tony Rice graduated with a BA honors Business Degree from Manchester. He worked as a management accountant for the electronics companies, Pye/Philips and Sinclair in the UK. In 1981 he came to South Africa to join Deloitte and Touche Consulting, and later moved to KPMG Consulting as a Director.

Since 1990 Mr. Rice has operated as an independent consultant specializing in production scheduling and supply chain integration.

He can be contacted at e-mail address: Production-Scheduling@Mweb.co.za
or through us at webmaster@symphonytech.com