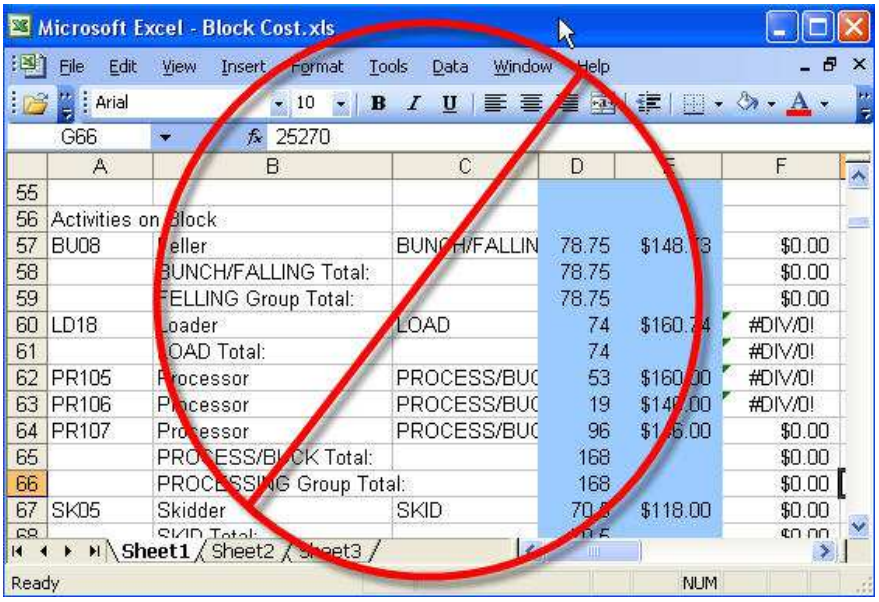
Just Say No to Spreadsheets

By Bob Lucke, Caribou Software Inc.

I'll start off by showing my age. I used the first spreadsheet package VisiCalc on an Apple II in 1982.[1] VisiCalc started the revolution in the office that has seen the 15-column pale green ledger pads and #2 lead pencil go the way of the check protector and the cha-ching of the old cash register. Since then, spreadsheet programs such as Lotus 1-2-3, Quattro Pro and the ubiquitous Microsoft Excel have established themselves as core applications in any accounting department. Easy-to-use and flexible, spreadsheets have been applied successfully to a wide range of business applications – from simple job estimates to complex financial simulations. Spreadsheets are often used to fill in the gaps where core accounting systems, such as QuickBooks, Peachtree, or MAS-90 fall short.



Logging companies use spreadsheets for many uses, including tracking load tickets, paying log haulers, calculating land owner payments, and building up cruise estimates and bid prices. Indeed, spreadsheets have been adopted for duties far beyond their practical use in logging operations. Often they have become so large and complex that they have been stretched to the limit. Herein lies the rub.

> *"The repercussions for a logging operation overly reliant on large and complex spreadsheets can be quite serious."*

The downfall of spreadsheets is that as they grow in data size and in complexity, they deteriorate. Large amounts of data make spreadsheets unwieldy: calculating and sorting takes longer, even just loading a large spreadsheet is a chore. In order to reduce the size of spreadsheets, some users split them up into multiple spreadsheets, which solves one problem only to create another problem: namely that of keeping organized a set of worksheet files. More and more time is spent updating and maintaining spreadsheets. Adding new functionality or even just adding columns becomes increasingly foreboding and perilous. And, woe is the company whose spreadsheet creator leaves for greener pastures and the spreadsheet must be taken over by someone else. Most spreadsheets have little or no documentation and only the original spreadsheet architect's brain contains the structure and sequence of steps required to manage the worksheet. Anyone who has had to decipher a

[1] As for a little bit of trivia: If you ever wanted to know why in Excel that columns are letters and rows are numbers, credit goes to VisiCalc that utilized the A:1 terminology for cell references. You can also blame VisiCalc for references that use column/row notation when every math major knows that cells are always referred to by row/column notation.

complex spreadsheet created by someone else knows the painstaking work involved.

The repercussions for a logging operation overly reliant on large and complex spreadsheets can be quite serious.

(1) **Payment Errors**. Bad cell references. Errors in formulas. Jumbled data from bad sorting. Lost data. Errors in pay not only have significant financial effects; they are also embarrassing and can irreparably harm your company's reputation. The people that work for you need to have confidence that they are being paid correctly.

(2) **Time**. Perhaps the greatest liability with complex spreadsheets is the hidden cost of wasting valuable time and resources. Wasting your own time – or your office staff's time – constantly re-keying, validating, manipulating, and consolidating data is time that could be much better spent in managing your company's operations. Admit it – you would much rather be outside in the woods than in some stuffy office figuring out why a spreadsheet doesn't tie to the GL.

(3) **"Lost" Reporting & Analysis**. So you've finished this week's contractor pay. You now want to calculate your costs for a specific job over several weeks. Or suppose you want to check your production against the cruise for each of your current jobs – and you have multiple jobs in progress. Or suppose you want to compare your accrued revenue against your latest mill settlements. Or suppose you want to match up your employees' hours by logging phase for each active job. It's not that you can't use spreadsheets for these tasks; it's that the work involved in doing these tasks on a regular basis is so large that these tasks are simply skipped altogether. The data are all there, but are not used effectively.

Don't get me wrong, I think spreadsheets are great under certain circumstances. They are transparent: you can easily check the calculations in any cell and you can trace any calculation to its source data. They can handle intricate calculations. They are easy to use. The problem, of course, is that these advantages have made people so comfortable with worksheets that inertia sets in. They are reluctant to give up that which they know and with which they are familiar. For many, it is hard to resist the temptation to build a new spreadsheet for every conceivable application. The hazard in this approach is that when your only tool is a hammer, every problem looks like a nail. The end result is a common situation where spreadsheets are all too often used (or misused) for applications beyond their capabilities.

There are Alternatives

Over the last several years, a number of specialized software programs have been developed for the logging industry. In general, these programs serve to bridge the gaps between Excel worksheets and standard accounting packages. These packages are designed to handle the business rules that are specific to logging contractors, yet are designed around a core database, such as Microsoft Access or SQL Server.

"It's not that you can't use spreadsheets for analytic reporting; it's that the work involved in doing the reporting on a regular basis is so large that it is simply skipped altogether."

What does a database application have that Excel does not?

Database applications have several virtues that provide huge advantages over Excel.

(1) **Single point of entry**. Data in a database is stored in a single place, but can be used for multiple purposes. For example, load tickets are stored in only one place, but can be used to create contractor settlements or do production reporting. The single point of entry avoids the copying or re-keying of data and its associated pitfalls. In a database, if a load ticket is changed, that change is reflected in all places where that ticket is used.

Thus, you can have literally hundreds of reports or statements that draw on the exact same load data.

(2) **Data Validation and Cascading Updates**. In a database application, validation lists are created in advance of data entry. For example, in a database application you would set up a list of valid species, say PINE, OAK and POPLAR. Then, when tickets are entered, one of these valid species must be used -- there is zero tolerance for typos or creative spelling. A load ticket will not be admitted to the database unless all entries are valid.

"Database applications have several virtues that provide huge advantages over Excel."

The benefits to this approach extend further. Let's say you have entered 800 load tickets, but then decide that you really want to use the term 'RED OAK' instead of just 'OAK'. In a database application, you simply go to the validation list for species and change the word 'OAK' to 'RED OAK'. This one change will automatically update all your load tickets that have a species of 'OAK' – you do not have to sort and copy hundreds of rows in a spreadsheet.

(3) **Scalability**. All data is stored in the same database file. For example, all load tickets, employee hours, cruise data, payment schedules, etc. are all stored in the same file. There is no proliferation of worksheet files. Moreover, the data storage in a database is practically unlimited; you can have thousands of load ticket records in the same database. In working with these thousands of records, the actual program selects only those records needed for its immediate purpose, unlike Excel which loads up all your data at once.

This scalability allows you to store multiple years' of data in one place and provides a basis for all types of reporting. Writing reports that slice and dice the data in different dimensions is a breeze. For example, you can produce production statistics, by tract or job, in one report; you can then turn around and summarize your pay, by contractor or landowner, simply by running another report. In addition, extending your application to handle new reports or additional functionality, such as overload tracking, is very often a simple task. Once the data are stored in a database, making use of them for any other purpose is usually straightforward and inexpensive.

(4) **Automation**. A database program works the same way all the time, as compared to spreadsheets that often require significant human interaction. For example, in looking up an applicable pay rate, a program always returns the same pay rate based on the attributes of the load (species, product, destination mill, etc.). In a spreadsheet application, users often manually select the correct rate to apply, opening up a window for human error to creep in.

Return on Investment

So what's the downside of a database application? I can name two: functionality and cost. Obviously, if a database application still leaves large gaps between your business rules and its functionality, it is not likely to be appropriate. For example, if you need the application to handle time cards and it does not, then clearly there is a major shortfall in functionality. At this point the cost of the application is usually irrelevant.

Once you are satisfied that a database application has most of your required functionality, the next issue is to determine its cost and benefits. The costs of adopting a new application are twofold: (1) the out-of-pocket costs for licensing the application and (2) your own costs of getting trained up.

The cost of licensing an application varies by software vendor. For the sake of argument, let's assume a logging-specific application costs $6,000. In our experience, it takes a new user around 25 hours to become really proficient with the software. If we assign a value of $20/Hour, the implicit cost of the user's time is $500, yielding a total cost of around $6,500. On its face, this appears relatively expensive compared to Excel or QuickBooks.

"The pay-back period based on these simple calculations is less than two years – certainly making it an attractive investment."

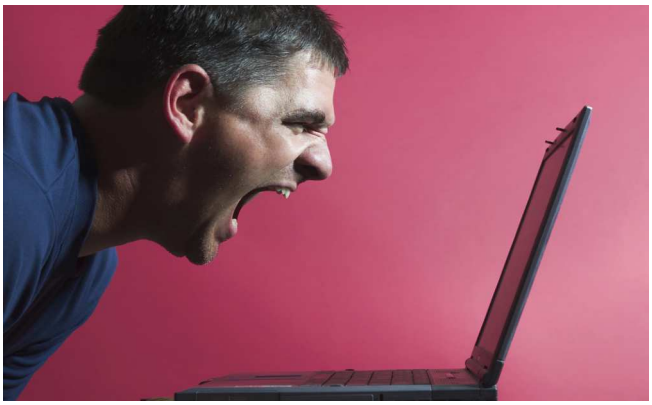
But, what are the benefits? Again, based on our experience, users should expect to gain around 4 hours a week purely in time savings. This translates into roughly 200 hours for the year and is an annual savings of $4,000 (again based on the $20/Hour pay rate). The pay-back

period based on these simple calculations is less than two years – certainly making it an attractive investment.

If this simple calculation is not compelling by itself, bear in mind that the benefits from a database application exceed just those in straight time savings. The value of reduced errors is not readily quantifiable, but certainly can be considerable. Moreover, the business efficiencies gained from enhanced reporting and analysis depend on each company's specific circumstances and in many cases can be much more valuable than the time savings.

Spreadsheets Still Have Their Day In The Sun

Not all data applications will cover all of your business needs. It is my experience that many loggers have special business rules or reporting needs that cannot be handled by any given software package. In these cases Excel and other spreadsheet applications still have a major complementary role to play.



The key is integration. You must be able to draw upon the data in the database application and use it as you see fit in your spreadsheet application. Any database program worth its salt should provide the ability to copy-and-paste data from the database application to a spreadsheet or at least allow you to export (and then import) your data. For example, you should be able to copy-and-paste a selection of load tickets from a database application into your own spreadsheet.

One drawback with cutting-and-pasting between a database application and Excel is that the data loses its integrity. That is, if the database is changed after a copy-and-paste, the spreadsheet data are no longer consistent with the source data, and you have to copy-and-paste all over again. One solution to this problem is a little-used (and somewhat complex) function in Excel that allows you to set up a database 'query' and import data directly from an Access or SQL Server database. A query is a set of instructions to the database that defines a set of records to retrieve. For example, a query might select all load tickets from a given tract or job delivered to a certain mill over a certain date range. This query can be stored with the worksheet and be refreshed at any time, thereby allowing your worksheet data to remain consistent with your database data. One safety feature with this query approach is that the data are 'read-only' meaning the spreadsheet user cannot alter the data, but he or she can always obtain the most recent data from the database.

Integrating your worksheets with your database application allows you to combine the best of both worlds: you can retain the flexibility of Excel and the ability to define your own reports or set up your own analyses, but still retain the data integrity of a database application.

The moral of the story is that you don't always have to say no to spreadsheets – you just need to know when to say no.

"The moral of the story is that you don't always have to say no to spreadsheets—you just need to know when to say no."