Donuts for DBAs IV – Documentation

By Redmond Bim.

Rather than my usual diatribe about the incompetence of most managers in the IT industry I thought I would look at a forgotten and rarely covered topic on the relevance of documentation in programming code. Though I do suspect that the usual readers of this column would like me to at least have a go at managers, I initially refrained. On review the article seemed to lack some of the usual lustre so I thought I would add the following paragraph.

If you are a manager in IT (and for those in management at the moment who are reading this replace IT for someone who deals with computer people. This article was most likely placed on your desk by a surreptitious employee who is now carefully peering over a partition looking to see your reaction, wondering whether you will even understand the jibe behind this article) and you know you are not comfortable at your job, or are suffering from the Peter Principal **, then leave and move onto bigger and better things. The vacuum that is left behind by your departure is bound to be more efficient and definitely more cost-effective than if you stay. Improve the quality of the IT management pool by leaving and ensure a better standard is maintained. There is a statistic I remember reading last week that stated how 40% patients died less in the week of a doctors strike. Sometimes having less management is better.

Now let's get going on the topic of documentation in code. I stress in code and not general user or technical documentation, which is a separate issue and one which I will not discuss at this point. Not because I do not want to. I really do, I would really like to spend paragraphs going through the need for and against it, but having a reputation in a previous life for my documenting capabilities I thought it prudent to leave it to another Redmond to cover this point.

There is a need to document code, but that need has changed. In my early junior days as a programmer, it was imprinted on my mind to comment every one or two lines of code. At the time it was stated it was needed for maintenance. Though I was writing in COBOL and as I had to write additional user and technical documentation on what I built, I didn't realise that the lofty goals bestowed on me by my work colleagues was just a belated attempt at keeping me busy and slowing me down from programming at a rate that would have left most of them way behind.

The tactic worked and at the end of three weeks programming I produced a well proven and working program, composed of about twenty lines of serious code with at least twenty pages of supporting documentation. I am sure that on the completion of my tour of duty the program I wrote was comprehensively maintained by a vanguard of studious application developers, burning the midnight hour scouring over every line of documentation written and using it to enhance the application.

Instead, I imagine on needing maintenance the program was deleted, rewritten and the documentation consigned to a dust bin (in those days recycling was only for aluminium cans and paper did grow on trees).

Based on years of experience and writing hundreds of thousands of lines of code, the one thing I have learnt is that documentation in a program actually hinders maintenance.

Some points to stress based on this experience:

1. If external documentation has already been written, why reproduce it in the code? Because that is how it always has been done, and been done since code was first written. Which is why I raised it, and believe this concept should be thrown out the window.

2. Comment Maintenance. If you put documentation into the code then it has to be maintained. One change to the logic and the comments have to be updated. So twice the work has to be done. As well as this, realise that programmers do not like writing documentation, it slows them down and the documentation then produced is of no benefit. So I ask the question, why force them? Oh, let Redmond answer this please – simply, because management want paper to justify a projects existence.

3. Comments make it harder to read and find information, thus making it harder to maintain the code. There is nothing more frustrating than wading through pages of code only to have to scroll past pages of comments proclaiming each procedure and function with author and modified dates, and what the procedure is for, why it was created in the first place, its history, its point of existence and what the programmer had for lunch that day. It's a waste of time. Yes it is. Believe me, it is. If you don't believe me you probably haven't programmed much before and you are in that category where you think you can program but need to be hand held through the coding process. Comments like this are the ads of the programming world.

4. Use your screen. There is nothing more frustrating than maintaining code built for a 800x600 screen. Wake up, most developers are on 1600x1200. Use the screen to code in and stop putting in all those useless line feeds. This has nothing to do with comments, but it aggravates me enough to want to raise it.

The amount of documentation placed in the code can depend on the coding language used, but I follow the simple logic which is: If someone is maintaining the code they had better understand the language it was written in. If they don't they shouldn't be maintaining the code and will do more damage by changing it. If they are experienced and understand the code, then 95% of what is written should be obvious to them. For PL/SQL, which is where I am focusing this article on, the language is verbose and is typically coded with voluminous amounts of SQL statements. Its obvious what it does, so why comment the obvious.

The 5% of comments needed should be reserved for the not so obvious. The coding where something tricky was needed to get around a bug or needed to improve performance. If it's a comment it will stand out, it will be easy on the eye and quickly read and absorbed.

Most PL/SQL code is self documenting. It's a language which is verbose and unlike C, its very hard to hand optimise. This means that what you write reflects what the code does.

So when it comes to documentation in code, keep it simple and to a minimum and everyone will be happier and the code easier to maintain.

** Peter Principal is a term coined to make reference to a public servant who has been promoted to his/her level of incompetence. In today's environment the Peter Principal applies equally well to a manager in a commercial business and can definitely apply to a manager in a large organisation.

Redmond sometimes says things he regrets. In his last article, "An Interview with Redmond", he lost his temper when the interviewer grilled him about his background. He cut the interview short. His regret is that he didn't get a chance to say what he truly thought of the paparazzi press moguls that haunt his every move.

If you want to contact Redmond and either congratulate him on this article or want to know where he lives so you can frail him alive for writing such nonsense, then you are more than welcome to send an email to the publishers of this magazine. Under careful supervision these emails will be forwarded to Redmond's private email address. Suitable versed articles might appear in the next column.