

Extract from Jim Lee's 'midrange matters' column in iSeries365 and iSeries News UK

Midrange matters with Jim Lee: Application Modernisation – Crossing a divide

Jim Lee sounds a call to arms for iSeries developers

I have a view that the very people who supported the iSeries to its prior peak of success are now partially to blame for the stall in that success. The IT staff, the programmers, the systems managers, and the systems designers in AS/400 installations are, inadvertently, a source of problem.

When you have been cosseted and protected in the wonderful world of i5/OS and (and it is an important and) you have been developing linear logic in such as RPG IV, then the world of servlets and applets is pretty scary. Your environment moves from rules and linearity to multidimensional chaos. It becomes too easy to desert the responsibility for you to change and, instead, to recommend a different platform that is managed by different resources including different programmers. After all, you are securely linked to the back-office, it is business critical, and these other guys disdain RPG. It appears to be a gamble worth taking; make it somebody else's problem. I believe that this is happening, all too often.

So, we see instances of i5 in the back-office surrounded by Wintel and Unix servers to deliver "Web based communications and services". Of course, this is entirely unnecessary. In fact, I think that it is meaningless. There are web servers for iSeries. The IBM Websphere family runs on iSeries. There is Java for iSeries. It cannot be the marketing pressure of competitive vendors alone that is creating the success of comparatively unreliable and insecure platforms. There have to be other factors at work leveraging this circumstance.

There are other elements of confusion for both technicians and non-technical executives that can be identified. 4GLs, Toolkits, and Middleware vendors all strive to pronounce their unique selling points for "Application Modernisation". When push comes to shove in the delivery of these components, there is one painful truth. Many



3GL programmers don't have what it takes. They don't get it. They have been developing linear commercial logic insulated from even the basic complexity such as record locks, and hardware failure. Even these 4GLs and tools cannot insulate the 3GL programmers from the new world. I appreciate that you can push RPG through Microsoft .Net and get C# out of the back. That is semantic. That is about syntax. The problem is construct.

The programming constructs encouraged by years of language such as RPG, Cobol, and Fortran have allowed programmers to develop code only for an organised and serially sequential world. "B" must follow "A". It is not possible, in this construct, for there to be no "A". And, "Z" is beyond credibility. Very few of these programmers (who have experienced only 3GLs) can cross the divide; write programs to expect the unexpected. Event driven logic is required. Events can be conditioned on what went before but the next thing to happen cannot be compulsorily predicted. Hence, the new style is composed of many smaller programming components that are independent and interdependent at the same time.

If a program is interacting with Browser content, there is no way that that linear logic will service this. The user will navigate as he wishes and the program logic, if it is to work, cannot refuse to track with the user. The program(s) can refuse the result or deem that there is no result. But it needs to be a different kind of program. It is not going to be the same program that was written for a dedicated terminal with a disciplined user.

My point is that success in modernisation projects needs a different kind of programming and that means different programmers. Many programmers will, and can, re-train. Regrettably some cannot.

What of the effect on iSeries? As I claim in the opening, the issues are being dodged. Instead of tackling the problem which exists for all and any platform, the option taken is to switch the new requirements to another platform and require different resourcing and management. We cannot allow this fifth column to continue. We have to stand up and accept that resistance to change is a powerful human motivation. The "New" programmers are required whether on one platform or



another. Concealing “Failure” by hiding behind platform means a less secure future for all of the players. Retraining the “Old” or recruiting the “New” has the same effect whether the modernisation takes place on i5 or its web serving competitors.

We have white papers and case studies that prove i5 consolidation, and i5 web serving, and i5 Workplace, all have been delivered on time and within budget. The cost of ownership benefits of staying on consolidated i5 carry over. Staying on, and expanding the use of, the i5 platform provides greater security for every player; programmers, managers, and the customer company as a whole.

If you are one of these programmers who find the new world of such as Java so terrifying, then accept that you are one of many. You may jump the divide (it is fun when you do). If you cannot, then imagine this; the new guy is coming anyway. Get him onside. Form a team. He is going to love i5. He is scared too. But, nobody told him in school that there was a completely defined Operating System. He has never experienced a world of ease of use and diagnostics. He will never have seen a platform that can act as a master console to a range of Windows Servers with dynamic disk allocation. Oh, and the masculine includes the feminine in all of the above.

Please, programmers, program managers, development managers, don't sit back and allow, and certainly don't encourage, that other servers are necessary and other resources are needed because you are not “Javaman”. There are thousands like you and too many of them work with divided teams competing for budget, all because of a, relatively, easy to understand embarrassment.

In the evolution of IT, this is not the first time this kind of problem has arisen. When I started my professional career, Assembler languages were the vogue. Tight code was written for very small memory. Every error from every device had to be handled in line. Along came 3GLs (COBOL first). The hype that was broadcast referred to speed to delivery of new code, and ease of maintenance. Nonsense at first. Assembler programmers used to deliver more than one program a day; usually right first time. Ease of maintenance was not attractive, it meant that just anybody might change the code. What, in the end, defeated all of the conflicts, arguments, and



Midrange Matters

www.campbell-lee.co.uk

discussion was Operating Systems. OS's assumed control and handled all errors and controlled the runtime environment. Now, 3GLs compiled code tightly linked to the OS. Assembler had little or no advantage. Some of us believed that the art had died and the artisan had taken over.

Defeat was not admitted in any short period of time. For years, the Data Processing Department (it was thus that we were known) defended itself with all sorts of "More for less" presentations to the company board. Resistance was futile. Operating Systems meant multi-programming and stability. Operating systems became essential for supporting "Screens"; and everybody wanted these.

The early days of "Screens" brought about a rejuvenation of the assembler programmers. "Do forever" polling and "Undo" held no mystery. Re-entrant code was a concept dear to their hearts.

You don't have to understand the mysticism of flashbacks. You need to understand that you know what you know and that is probably a very great deal. You need to accept that change is underway. You need to accept that the effects of change are permanent. Even if you can't change you will still contribute.

Please stop hiding behind the Somebody Else's Problem option. Stand up. Be counted. You do not need to desert the best operating system ever developed (and about which you probably know a very great deal) to be right at the forefront of today's commercial technology.

ENDS - 08/06/05