What Makes a Great E/CTRM? Part 9 – Technology

Bob sat in a large conference room while several of his team streamed in slowly through a door. Please close the door, Bob said to the last team member to file through the doorway. Bob scheduled this meeting with his own technology team to review technology requirements for a C/ETRM system Bob's company was proposing to build or buy. If you missed Bob's meeting with his Credit Management group, you can find it here.

"Thanks for coming, everyone." Bob began. You all know our company is trying to figure out what sort of features and functions a C/ETRM system should have to meet our company's needs. I want to begin a conversation today regarding what sort of technology we should be looking for. If we could do whatever we wanted, what would we choose to do? If we built our own system, what technologies would we prefer? If we buy a system from a vendor, what would be our technological preferences? Bob stopped speaking and waited for his team to begin sharing ideas. In his mind, he could hear an imaginary voice, "Bueller. Anyone. Anyone."

After a moment, Sam raised his hand. Bob directed everyone's attention to Sam. Well, to begin with, most corporate systems today are built using Microsoft's .NET framework. It has a great IDE. And, C# is a pretty popular language. Sam appeared to have tossed his two cents worth into the conversation. He was done. Thanks, Sam.

Joy chimed up. I like the SQL Server database management system (DMS). It isn't very expensive, if we don't use the Enterprise version. Oracle is supposed to be the best, I hear. But, since we are a Microsoft shop, makes sense to use SQL Server, right? Everyone knew that each DMS required some time to build competency and familiarity. Who knew which of the two most popular DMS systems was the "fastest"? Ultimately, both would probably work well. The decision normally came down to preference. Thank you, Joy.

Another hand shot up. Yes, Doug? Doug spoke up. I think .NET and SQL Server make really good sense. Especially for backend computing. But I would like to vote for a more modern client application development language. All new client application development these days use HTML5 and Javascript, it seems. The reason is that these languages work on just about any device: PCs, Smart Phones, Tablets. I think our users should be able to work from anywhere at any time, right? Sometimes, people can't get to their laptop. But they will probably have the Smart Phone with them. And, I'd like to suggest we consider Java, too, at least for the client application. It is a really popular language and has been around even longer than C#. Thanks, Doug. Bob nodded in his direction.

Doug made a good point. If Bob had a choice of user interfaces (UI), why wouldn't he pick a more modern one? Why would he recommend his company spend lots of money on old technology? Whether the system was built or purchased, Bob's company was going to live with its new system for a very long time. If the company picked a vendor today, which sold older, less current technology, the technology was certainly going to be very old, possibly obsolete, in another ten some odd years when his company again looked for a replacement C/ETRM system.

And, who would be able to help his company to maintain older technology? Bob remembered Y2K. At the time of the turning of the millennia, lots of mainframe programmers were hired to fix old mainframe programs written in dinosaur languages like COBOL. The worry was that the original programmers had used 2-digit years in dates because it used less of the precious, expensive memory of older computing machines. Then, when the clock ticked around to the year 2000, the programs wouldn't function properly using just the last two digits, '00'. Finding COBOL programmers then was expensive. Job paid well. But immediately after Y2K, after the crisis, the paychecks went back to normal. And, everyone wanted work on newer technologies. Couldn't find anyone to take a job working on the older technologies.

Bob saw the problem would be the same today. Finding programmers to develop on old technology would be difficult for both Bob's company, if they built a system; or, if the company bought a vendor's system, the vendor may not be able to staff its own development at some point. Bob made a note to understand the technology of any vendor reviewed by Bob's company.

Bob's thoughts were interrupted when Jane raised her hand. Jane was an IT business analyst, who helped internal company departments with their various IT related needs. I think, Jane said, that whatever system we get, it had better help our customers do their jobs. It must make their lives better. Bob nodded. Jane's point was a good one. What's the point of getting a system, even with the latest techno-gadgets, if it didn't make the company more efficient and productive? If it didn't do something to improve some aspect of the business?

Bob had seen several boondoggle projects in his career. They always started off with good intentions. But, very soon, the projects got distracted with a goal to be the most modern and cutting edge. Bleeding edge, Bob thought. Haemorrhaging edge, even. Technology for the sake of technology. Oh, the project team had plenty of enthusiasm and esprit de corps. Project failure couldn't be blamed on lack of productivity of the staff. But the focus was all misdirected. In the end, the projects produced no worthwhile products for the intended business users.

Technology was supposed to be used for the sake of business profitability. The temptation, the lure of technology, for developers is just as powerful as the song of the sirens in Greek mythology. Bob knew he should avoid this sort of thing. It would be harder if his company decided to build its own C/ETRM system.

If that were the case, Bob's team would have to spend months, investigating technologies. The goal would be to select technology that would endure the test of time. Bob would want to pick a technology that became a standard. Not a fad, or a flash in the proverbial pan. And, after that, Bob's team would need to spend a great deal more time developing an application development framework on which a C/ETRM could be built.

Without one, Bob's developers would all do their own thing. There would be no consistency across the code. Can of worms. Or, spaghetti.

But if a vendor's system was selected, the technology decision would be baked into the cake, so to speak. You get what you get. The temptation would be different. His team may be lured to select a vendor with a great technology stack without regard to functional breadth for his company's users, putting Bob in a precarious position. Bob knew some balance was required in a final decision to select vendor software.

Thanks, Jane. Well-spoken. Anyone else? Bob waited. Single sign-on, someone belted out. Cloud-hosted, another voice joined in. Why cloud-hosted, Bob inquired? We have lots of computer servers here. Bob gestured at the computer room across the hall. Jill had been the one to suggest cloud-hosting. Jill agreed, we do have lots of computer servers. But the hosting sites have some nice features, such as scalable on-demand services. And, database services are also available. They do backups and everything. And, we could mirror our environment with a disaster recovery services. Takes a lot of the burden of supporting this application off our hands.

Jill made a great case for hosting the application in the "cloud", so-called. Bob wasn't necessarily giving up much moving in the cloud direction. And he would be avoiding the risks of a "down" application server, which is always politically charged. Bob could keep everything else, all his computer servers, just the way they are. Minimal impact to his operation and budget. And, his company wouldn't need to apply to the board of directors for a capital expenditure on new computer servers, which depreciate quickly. These hosting sites charge by the month, data space, and computing resources used. Operational budget. Thanks, Jill. Really good idea.

Bob scanned the room from one side to the other. Finally, Scott raised his hand. Yes, Scott? Scott began. Doug mentioned backend computing. I thought I'd add to his comments. Most modern product architectures are called N-Tier. We all know about the database tier; one or more application services tier(s), one or more client services tier(s), and finally a user interface tier, which is where that HTML5/Javascript could be used, right? We should probably make a note to avoid client-server architectures, which have just two tiers. A three-tier architecture isn't exactly modern, either.

Bob made a note to get information from each vendor about each product's software architecture. If Bob's team built a system, it would have the latest architecture. But, if the C/ETRM system was purchased, the architecture would be whatever it was. The user community would choose what they liked best. And, they would like best a system that helped them do their business. They wouldn't really worry about architecture, would they?

That was Bob's job. While he had a voice in the decision, his was just one voice. An important voice, to be sure. This architecture argument goes back to the question about acquiring older technology. Bob would need to communicate this argument in a list of all the pros and cons. And then let the chips fall where they may. This decision had to be a group decision. The alternative always ended in finger-pointing.

Well, unless anyone has anything else to share, Bob concluded, I think I have all that I need from this meeting. Thanks. Bob got up and filed out of the room with the others. He would summarize his notes and make sure these requirements were put with the others. There was one more constituency Bob hadn't interview: Management. Bob wondered what sort of requirements his management team would add to his list. More about this interview to come in a later instalment.