What Makes a Great E/CTRM? Part 3 - Best-of-Breed

In our last installment, we meandered through Bob's thoughts as he considered DIY, or scratch-building his own C/ETRM solution for his company. Bob placed these thoughts aside so that he could ponder the possibilities of what is commonly called "Best of Breed" solutions.

Immediately, Bob's imagination got the better of him. In his mind's eye, he envisioned a peculiar looking animal of the canine persuasion. Each part of this dog-like creature seemed to come from one breed or another, each part a different breed. It was all dog, certainly. Although Bob wasn't a dog person himself, per se, he could easily recognize which parts came from which breeds.

Bob shook his head. "Doesn't make sense," he said to himself. "Hard to believe such an animal could function well in the real world. And, it looks ridiculous." Such were Bob's musings, initially. Funny that a C/ETRM solution architecture would be called in such a way. Best of Breed. But that's what it is called: When the best parts of the best of systems are bolted together to form one complete solution.

On paper, Bob thought, the idea does seem to have some logic. After all, no one system will have all the features and functions required by Bob's company's management, right? Why not acquire the best parts of several? Then, somehow weld the best parts of each together to function as one complete solution. Ah, Bob thought. There's the trick. His English friend would have said Sticky Wicket. Bob didn't know what a Wicket was. Or, why it would be sticky. But he understood the meaning. How to weld systems together. Can it be done? Technically speaking?

Bob knew enough about databases and programming language to understand some of the complications. Is there a common database management system (DMS) among the best of breed choices? Bob looked at the brochures and marketing copy left by his recent vendor visitors. They say they are database agnostic. Bob shook off a rather funny thought, "Does that mean they doubt the existence of a database?" Or, "Whether there is a database or isn't, they really don't care." Funny terms. Who thought of this stuff?

Bob tried to get himself serious again. OK. I need to know for sure whether each vendor prefers one DMS over another. These sales guys aren't technical, are they? They probably couldn't say, for certain. But, if one DMS is preferred, it probably means the system runs out-of-the-box on that DMS. Getting it to work on another database, agnosticism aside, may require additional expense. Need to find out if my DMS preference is the vendor's DMS preference for each best of breed vendor. If not, I'll need to factor in additional project costs. First thing to check.

Bob sat back and stared at the ceiling. How to bolt one system to another. Next thing to check: Does each vendor's database have similar entity-relationship? (That's database table structure for the uninitiated.) Hmmm, Bob thought. Will each vendor share with me its database table structure? Before I buy?! That should be the next thing I check, thought Bob. Intellectual Property is a highly confidential thing. Will a vendor grant me access to their database before I buy a license for a product? What about after licenses are purchased? Good questions to ask.

Bob has a friend at another company. That friend had told him that his vendor wouldn't even let him have access to any database tables. All-access requests to data had to go through the vendor's API (Application Programming Interface). Would an API provide Bob with enough levers and switches to make an interface work between two very different best of breed systems? That is a really good question.

Let's assume a vendor does let me peek behind the Wizard's curtain, Bob considered. Are there any obvious integration points? This one was a head-scratcher. Bob knew that in order to pass information back-n-forth between systems, common unique identifiers (UIDs, for short) would need to be cross-referenced. What would be the most important integration point? Bob's first thought was Business Associate unique identifier or BA UID for short. BA means counterparty, company, or business entity with whom a company does business. And, this same code would be needed to interface with the most important of Bob's company's system, the financial accounting system. And, credit management would require the same code, too, to piece together a view of a trading partner's overall credit exposure.

All the other common codes needed in interfaces would depend on what features of which systems would be used in the final best-of-breed solution. For example, if one system is used to capture and value trades and report risk positions, and another system were used to schedule physical trades, then presumably unique trade identifiers, or Trade UIDs, would need to be shared between systems. Would a scheduling system be able to store Trade UIDs? Something to check.

Information should flow in the opposite direction, too. A scheduling system should return actualized quantities back to the trade management system performing deal valuations. Scheduling systems work normally on Parcels, so-called, Bob knew, which could be a small part of a trade quantity. Probably need to have parcel-level unique identifiers, too, Bob thought. Now the big question: What if the trading system doesn't have parcel UIDs? Or even the notion of a parcel? How will parcel quantities in the scheduling system get passed back to the trading system?

Bob began to wonder how many of these UIDs there were, and how many of the best-of-breed systems had each UID in common. Could he get that sort of information from his system vendors? Certainly, if they did have the UIDs, they would share that with him, to allay his concerns. But what if they didn't have the UIDs Bob needed? What would they tell him? Again, this intellectual property stuff is a touchy subject. Need-to-know stuff, as they say in government jobs.

All these things to organize and sort out. Just to create interfaces between systems not

designed to be interfaced. And, interfaces can be such tricky things to develop. Can a person anticipate all possible states of data so that an interface can be programmed to respond to irregularities in datasets? Will any important information fall through any cracks (unhandled conditions) in an interface? And, will each interface alert someone to problems needing attention? Bob thought. OK. An interface can log all data moving through it. That shouldn't be a problem. But, how will it alert me, or some other person, to resolve an issue logged there? Something else to ponder on.

Bob stepped back from the minutia for a moment. Development and maintenance of all these interfaces seemed like a rabbit hole. How much time would each interface consume to develop and test? How many developers? Did Bob's company possess the sort of patience, which may be required to persevere through the pains of rolling out an enterprise solution with these many potential points of failure? And, the cost of acquiring each vendor system, deploying it, training users, and performing the acceptance testing. And, all the while, building interfaces between each system, so that the interfaces are developed, tested, and in place when all the systems come on-line? Like changing wings mid-flight, Bob mused.

Then, there is the constant monitoring of interfaces, and patching code, and associated human resource costs. And, the disruption to on-going business, which must wait each time an interface goes down or data doesn't transfer correctly. How will system users know their reports do not reflect values, which are trapped within interfaces? Will they make trading decisions on imperfect data? Who will be blamed for that? A bead of sweat formed on Bob's brow. Bob realized the stakes were bigger than first thought.

As Bob's mind raced down this thought, he remembered that vendors periodically require their customers to upgrade to a recent version under support. What happens when one vendor sends a patch or an upgrade? Will that patch or upgrade affect any interfaces? Will it affect a downstream vendor system? How much time and cost will be required to roll out an upgrade and test everything again, especially the interfaces?

OK. Bob thought. Mixing best-of-breed systems to manage the life cycle of a trade wasn't going to work well at all. Just like having a dog with the head of one breed and the tail of another, mixed breeds probably wouldn't perform any function particularly well. But what if Bob used each breed for its intended purpose? Pure breeds. Hounds to hunt, greyhounds to race, chihuahuas to …, well, Bob didn't exactly know what chihuahuas were bred to do. But this idea had merit. One C/ETRM would handle one part of Bob's company's business, and another would handle another, along the lines of each traded commodity.

Bob leaned back in his chair, his hands locked together behind his head. That is a brilliant thought. And, very doable. This is an idea he could present to his management and company's staff. Now, what could be the complications? Think. Think. Think. Well, first of all, Bob realized that there would be more interfaces. Yes, multiplies of more interfaces. In fact, a set of interfaces for each different C/ETRM breed used for each part of his company's business. That thought cooled Bob's enthusiasm a bit.

Bob realized his staff would be required to develop some expertise with each C/ETRM breed, possibly doubling or even tripling his staffing costs. One can't expect a single person to know everything about every system, right? And, even if one person did know everything about every system, how much would that person cost to hire? And, there are only 40 standard working hours in a week. Bandwidth limitations.

And, Bob would need to hire at least two, just in case one resigned. Having coverage was just basic management, right? Or, another

thought, Bob could retain the services of a company having staff with the required skillsets. That staff will probably be a little more expensive per hour or day, possibly, depending on negotiations and contract terms. Contract employment, as opposed to permanent employment. Doable.

Bob concluded that while best-of-breed systems do seem to have some merit and logic superficially, he was concerned about the complications, complexities, and costs, which were potentially budget-breaking. Particularly and especially for mixed breed systems. But combining pure breed systems, however, just might work.

One last thought occurred to Bob, as he was about to take his victory lap. Reporting. How would Bob combine data from each pure breed system to generate executive level management reports? Data warehousing projects almost always turned into boondoggles. Bob's company couldn't afford that sort of waste. Bob would need to research some new technologies he'd heard about, called BI tools, which Bob surmised meant Business Intelligence. Another funny term, Bob mused. Possibly, a path forward, though.

Bob's eyelids grew heavy from the weight of these thoughts. He began dreaming about the perfect system, which didn't require him to develop anything. No interfaces. No warehousing reporting system. A C/ETRM solution addressing all the needs of his company both now and in the future. A master breed. Like a wolf. Aren't wolves supposed to have all the genetic material for the whole canine kind? A master breed C/ETRM all integrated in one architecture. BI executive reports. No internal interfaces. It was a good dream.

But, does such a solution even exist? If one did, Bob's life would be a great deal better off.