From: www.cio.com

Why CRM Data Trumps CRM Systems

– David Taber, CIO

October 26, 2010

At last month's CIO Forum in San Francisco, I lead a round-table discussion about the strategic way to think about CRM applications and their evolution in the enterprise. While everyone agreed that the most comprehensive CRM systems were built (actually, assembled), not bought, one CIO took it a step further. What really matters in a CRM system isn't the system at all. It's the data.

And in an enterprise of any real scale, that data won't ever live in any one system. It will be held in various CRM, call center, support, and Web systems. Even if these were all from the same vendor, the instances would be operating and managed fairly independently. Even with SaaS products (where by definition everyone is running the same code line), the data models and semantics used in the separate instances tend to drift apart. So the data will tend to be siloed unless there's strong governance to prevent it.

To misquote James Carville, "it's the data, stupid." CRM features are important for users, but what matters to IT — and the health of the business — is the quality, validity, and usability of the data held in the customer-related systems. From this data-centric perspective, there are a range of new criteria of how CRM systems should be evaluated.

Federated Data

As I wrote in July, in large organizations a decentralized approach is often needed for CRM systems. To make that practical, however, data must be accessible and usable from several of the federated systems. This means that the following must be available from CRM systems via APIs, Web services, or RESTful calls:

- Data values and data state (e.g., locked, stale, valid)
- Metadata (at least the object model and field constraints)
- Basic read and update operations (creates and deletes will probably require special privileges)

Not all access will be done via low-level coding, of course. The number and robustness of commercial adaptors is an important CRM selection criterion. Low-end adaptor vendors focus on ease of setup and low cost, and they are increasingly delivering their adaptors as a SaaS offering. For straightforward random access to records, the low-end approach works out well — but watch out for throughput surcharges that may be part of their SaaS pricing model. At the high end, the focus is on either throughput (for ETL/DW use) or transactional complexity (for two-phase commit, workflow, etc.), so vendors deliver either dedicated integration software or information appliances. For popular CRM systems, open source integration servers are quite viable, but they typically don't have connectors to down-rev CRM products or home-brew systems written in "old" languages.

As the data in a CRM system is put to more external use, almost inevitably the fields or objects will need to be modified. Extensibility of the CRM system's data model is a critical attribute. It must be possible to add fields, modify constraints, and add sub-types of records, even if it's not all done through the administrative GUI. Creating alias fields (updated through formulas or workflows) or morphed / denormalized objects (through code) will almost certainly be needed to facilitate federation.

A real challenge in federating data is the construction of historical views which are required to understand the behavior of customers and the effectiveness of campaigns over time. CRM systems store and present the most up-to-date customer information. I don't know of any system that is based on versioning and can natively give you the "state of play" at a point in time. Some systems do have change logs for key objects, but all too often that feature isn't turned on or is looking at a small subset of variables. Further, change logs are not structured in a way that makes point-in-time reconstruction

terribly easy.

Some systems do provide automatic periodic snapshots of key objects. In a federated environment, at least one of your CRM instances won't offer this features — so you'll have to do the snapshots through external code and store the results in a data mart.

Federating data may sound hard, but making it meaningful is the real goal. Most CRM systems have no data dictionary, and the semantics of certain fields and states seem to be folklore. As there isn't really a technological solution for this, policy and procedure are the answer here. If you already have a data warehousing project covering some of your CRM data, build on the knowledge base it has built. Following the spirit of federation, your data dictionary should be built as a living document, crowd-sourced, and held in a Wiki. Since the data dictionary may contain some privileged information, make sure that the Wiki requires a login (at least for writing).

Looked at through the lens of data federation, everything looks like a database. A CRM system with the coolest UIs may hide a nightmare of data that's not available through an API or is inscrutably written across 250 tables. Make sure to include the "federation friendliness" as a criterion for any system you're considering for enterprise use.

David Taber is the author of the new Prentice Hall book, "Salesforce.com Secrets of Success" and is the CEO of SalesLogistix, a certified Salesforce.com consultancy focused on business process improvement through use of CRM systems. SalesLogistix clients are in North America, Europe, Israel, and India, and David has over 25 years experience in high tech, including 10 years at the VP level or above.

Follow everything from CIO.com on Twitter @CIOonline, and the CIO.com Facebook page