From: www.cio.com

Expand Your CRM System: 3 Paths to More Power

– David Taber, CIO

July 20, 2010

Somebody brilliant once said that CRM systems are frameworks disguised as applications. Many an analyst has said that the best CRM systems are built, not bought.

So it shouldn't be too much of a surprise how lean the feature set is in CRM systems — particularly those deployed as hosted solutions. There's absolutely nothing wrong with deploying a fairly light set of features sooner, so you can start the ball rolling for the all-important user adoption. In fact, that's what we recommend.

But if you're successful, the users will clamor for more features — both depth and breadth. So your imperative will soon be to get the new functionality deployed while the iron is still hot. What's the best strategy?

The answer, of course, depends on the scope and depth of the feature gap. Going beyond the configurations and formula/workflow items that your system administrators should always be doing, there are three main paths toward expanded CRM functionality:

Free / Open Source Plug-Ins

Every CRM vendor develops an ecosystem of developers around them, and the health of this ecosystem is important to you even if you never intend to use the free stuff. More dynamic ecosystems mean a bigger pool of developer and administrator talent. These will be important to your project staffing, whether you plan to use consultants or hire.

For sheer numbers, Salesforce.com has the strongest set of freebies, with functionality ranging from simple templates to complete applications. They wrote about half of their free/open source add-ons themselves — a considerable achievement. Beyond numbers, you need to evaluate the user feedback scores (hopefully, there's a "star" system) and how recently the plug-in has been updated. Since even the most robust CRM platforms evolve, it's important to see how actively the community is keeping the freebies up to date.

The usual disclaimers about evaluating and using open source software apply, even if the plug-ins are just freeware. Support and training are almost never included, and sometimes not available at all.

Add-on Products

Every CRM vendor also develops a commercial ecosystem of products that range from platform extensions to reports to administrator tools to adjunct applications. In the CRM world — almost irrespective of vendor — the most vibrant areas are marketing automation, system administration, data quality, reporting/analytics, integration, and telephony support (both PBX and mobile) products.

For sheer numbers, Salesforce.com again has the strongest story, with several third party vendors competing in all of the key categories. As before, you need to evaluate the user feedback scores (hopefully, there's a "star" system) and the buzz in user discussion groups.

The market has been evolving so rapidly it's important to discount reviews/feedback from more than 2 years ago. Indeed, the competition has been so intense that a couple of leaders from back then have practically fallen apart. Don't be too picky with the vendors about references — if you're too narrow about whom you're willing to talk with, references are surprisingly hard to find.

Some of the more ambitious and specialized third-party apps are surprisingly expensive. The laws of software economics still apply to the SaaS world: low volume products cost more to make and

support. It's nice to know that some things don't change.

Ease of Development / Power of Tools

Of course the final alternative is DIY. While the positives and negatives of development are fairly straightforward, what's different in CRM is the range/scope of development projects. It's surprising how often you'll need to write a trigger, stored procedure, or code fragment for things you'd think would require no code. Indeed, some on-premises CRM systems virtually require you to write XML, scripts, and SQL just to get a new report. So the projects will be small, but frequent.

For systems like Salesforce.com, SugarCRM, and Microsoft Dynamics, coding is explicitly encouraged for everything from screen widgets to triggers to business logic. Salesforce has proprietary dialects of Java and XML that are pretty easy to pick up, and they've produced a development, test, and deployment environment that is pretty rich. SugarCRM is able to leverage the PHP community for libraries and development tools, and Microsoft of course has an even stronger development and deployment story. You even get your own set of 174 new DLLs to work with.

The key risk of CRM development projects is that they are typically small and relatively independent of each other. While that makes them easy to get into production, it means that the developer talent drifts on to other projects fairly quickly — after a while, learning curve effects start to take their toll. Even if all you have are "lite" feature requests, the accumulation of code fragments and customizations can become quite a challenge to manage — precisely because they are scattered like band-aids all over the system. Make sure your administrators keep detailed logs of all their CRM modifications in a Wiki or other content management system.

This goes double for integration (data synchronization) projects, which are almost inevitable for a CRM system of any size. The development will be mainly scripting, simple queries, and workflow...with the occasional compensating transaction when rollback doesn't work. Since the niggling details of integration are subject to change whenever any of the systems involved undergoes change (a new table or a software update), you'll need to have good documentation of how/why the integration works, as well as test data to make sure that no corruptions have been introduced by change.

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.