COMPUTERWORLD

Defeat The Hidden Enemy of CRM Projects: Scope Creep

David Taber

July 27, 2009 (CIO)

Any big software project is vulnerable to the evils of scope creep. Project estimates are wrong, new requirements are added, and the next big bang release falls farther off schedule and out of budget. What are the specific tactics that can lower the impact of scope creep in CRM systems?

CRM systems are different from most other enterprise applications. While some requirements are non-negotiable, there is far more room for redefining requirements and deploying functionality in an incremental way. This means CRM systems can use a "by-the-drink model" of investment, rather than the "bet-the-farm model" characteristic of traditional enterprise software.

SaaS based CRM systems give you even more flexibility, as there are no hardware or heavyweight operational issues to deal with-scaling up or turning modules on is just a phone call away. CRM systems with solid web-services APIs such as Salesforce.com's make for even more flexibility in development and deployment cycles.

Here are tactics that fight scope-creep, borrowed from several disciplines.

Deliver small units of value to the business

Working incrementally isn't a bug-it's a huge feature of modern CRM systems. Thanks to the loose coupling of web services, there is every reason to keep functional deliveries small, simple, and separable. Sometimes highly desired feature enhancements take only a few man-hours to develop and test, so deployments can happen twice a quarter or even more frequently.

Taking this idea further, it is sometimes desirable to deploy only a portion of functionality even if an entire feature set is ready. Keeping features in reserve can ease learning curves, acclimate the organization to smaller and more frequent deliveries, and provide IT with "dry powder" for the next deployment cycle. (The ultimate weapon against scope creep is delivering something!)

Push back hard on requirements

Since the first line of defense against scope creep is avoiding spurious requirements, use the philosophy of Agile projects (even if you don't use the specific techniques). The user may not really know how important a requirement is. And even if they're adamant about a feature, they can rarely quantify the business value of having it. So get a core system-essential functionality onlyup quickly and add to it as the business case for "the next feature" becomes crystal clear.

Where this tactic runs into trouble is when your new CRM system is replacing an existing one. User expectations will have been set by the previous system, and they'll insist that they can't live unless the previous functionality and all data from the previous system is available on day one. This sets the stage for a "big bang" release and practically guarantees scope creep. It may take some hard arguments at long meetings, but you have to get across:

• Do you really want the new system to have all the data quality and functionality problems of the

old one from the outset? Isn't the whole point to do better?

• Users and customers have become used to screens, reports, and functionality of the existing system, and the transition needs to be as smooth as possible. A "slash cut" from one system to the other is likely to be disruptive, while a staged, incremental switchover will be less jarring.

• The data in the system-transactions and evolving customer relationships-must not be put at risk. The core data and functionality must be stabilized in the new system before it can be the foundation of the higher-level system functions. Some period of either "parallel play" (redundant system operation) or diminished functionality is inevitable and should be built into the plan.

• The impact of a CRM system depends upon users actively leveraging the system to collaborate around the customer relationship. If the users aren't there, it doesn't matter if you add more functionality-there won't be more business impact. Furthermore, adding more (complex) features to the system may actually work against usability and adoption. This user adoption argument can become a gating factor in controlling scope.

Call their bluff

Some requests for user features are simple and straightforward to implement. Other "simple things" can involve expensive data rework and external system integration. When you suspect a big "nice to have" in the feature list, use budgets to call the requester's bluff. Create a good cost estimate for that unit of functionality, and tell the requester that work will start on that element as soon as they provide the budget. Don't start work until they provide a specific charge number or inter-departmental budget transfer.

Re-prioritize CRM requirements every six months

In some industries, the half-life of a VP of Sales or Marketing is 18 months. The priorities set at the start of a CRM project may no longer be relevant due to internal policy or business-rule changes. Further, the evolving reality of competitors, partners, and customers can mean significant changes of emphasis in CRM and surrounding systems. In these prioritization sessions, compare features on the basis of true costs and business value. Avoiding requirements that are no longer critical is one of the best ways to keep scope creep at bay.

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.

Do you Tweet? Follow everything from CIO.com on Twitter @CIOonline.