Print Article Close Window

From: www.cio.com

# Why CRM Implementation Needs Training Wheels, Not Racing Gear

– David Taber, CIO

**August 13, 2013**

The most basic CRM implementations can be done as a "simple" package deployment, with no custom code and a couple of simple plug-ins to a standalone system. But that's just the basic ones.

More ambitious CRM implementations require more ambitious customization and extension. It doesn't take long before you have to think seriously about the development strategy you use. Making the wrong choice has serious consequences in costs, data quality and user satisfaction.

**Analysis: How Much Should You Spend on CRM Software?**
**Also: 4 Affordable Small Business CRM Options**

The specifics of this topic vary significantly for each CRM vendor. To keep this article short, I've focused on Salesforce.com, as covering all major vendors would require way too many pages. But the general principles explained here apply to all vendors.

## Take Off Training Wheels At Your Own Risk

Salesforce provides a wide array of features that are essentially "training wheels" for early, quick deployments. The features are easy to understand, require zero coding and are (more or less) user friendly. For a CRM implementation with only a couple use cases, these features may be all you ever need.


Unfortunately, these training wheels are dead wrong for anything sophisticated. An easy example is using multiple lookup fields on the same object. One example is having fields for Board Member, Attorney, and Auditor on an account, all pointing to contacts. It's easier to understand and report on than

using a junction object such as "influencers," which allows for an arbitrary number of people (of any job title) to be added to an account. Other common examples are using lead sources versus campaigns, or sending lots of emails from workflows. These are bad ideas all the way around.

**Related: 10 CRM Mistakes and How You Can Avoid Making Them**

The most critical items for review are validation rules, workflows and flows that users may have set up. These bits of automation will seriously conflict with custom coding. The two big areas of conflict with these technologies are the inability to fully control sequencing (leading to race conditions) and erroneous or misleading error messages (leading to chasing-your-tail scenarios). I won't go into the specifics; just know that when you bump into these issues during debugging, you'll start swearing.

The solution: As soon as you start doing any juicy Apex and VisualForce code for an object, replace all the workflows and validation rules for that object with Apex classes. This, along with all the test code you'll have to write, can be a major undertaking. But it's in your future if you do serious development.

You can take it one step further, too. Salesforce's best practices guidance states that there should only be one trigger per object. All the work of the many triggers you may have written gets moved into supporting classes, and all the test code for these classes accordingly gets moved into its own special test classes. This leads to a major code refactoring exercise, though Eclipse can help (somewhat). As you curse the hours you'll spend, just know that this is a "pay me now or pay me later" situation.

Now for the *coup de grace*. Remember the old days of error logs that were actually useful in debugging production problems that happen in the middle of the night? Those logs are just a fond memory unless you build code to capture error conditions and generate them yourself.

**More Tough Stuff: Email-to-CRM Contact Connection Easier Said Than Done**

Salesforce, like nearly all cloud-based applications, has training-wheel logging. For performance and storage reasons, the logs are recorded only when explicitly requested—and they stop after a few transactions. Since error conditions are the most important log events, the logs should be stored outside the system, pushed out by some nice Web service classes. I've been looking for a Salesforce add-on for this, but I've yet to find a good one. This is an area where system integrators have to develop their own toolkits.

## Get Back to Agile-First Principles

In IT, coding is second nature. Salesforce is a well-thought-out platform for building business apps. So knock yourself out for features that are absolutely required.

Reread the word *absolutely* in that sentence, because you'll be tempted to add some features that really aren't critical. In the land of Salesforce, features aren't as important as data quality. Features aren't as important as flexibility. Features—unless explicitly requested by a lot of users—get in the way.

**Commentary: Why 'Agile Project Management Controls' Isn't an Oxymoron**
**More: If You Can't Go Agile for Your CRM Project, Fixed-Price Can Work**

How? Ongoing flexibility is a critical success factor in CRM systems. This lets them accommodate the inevitable changes in requirements over years. Every feature you code in Salesforce will reduce the system's flexibility in the objects you touch and, perhaps, in related objects. Too much code means the users can't even add pick-list values without a complete coding, testing and deployment cycle. If you don't build your code in some really clever ways—using lots of introspection and dynamic techniques, for example—the tiniest changes made by users can have big consequences.

Even if you're coding in the best possible way, the code from third-party products may not play nice. Every time they upgrade, and every time you make a change, there may be hundreds of test failures and new anomalies to troubleshoot.

You really have to apply the fundamentals of value engineering and agile. Keep the training wheels on as long as you possibly can—and when it's time to get serious, create just the essential features with the best coding practices you can. Then you can add the Shimano and Campagnolo racing gear.

*David Taber is the author of the 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. Taber has more than 25 years of experience in high tech, including 10 years at the VP level or above.*

*Follow everything from CIO.com on Twitter @CIOonline, Facebook, Google + and LinkedIn.*