

**An Industry Insider's Survival Guide  
to Everything You Should Know About**

---

# **BUYING AND IMPLEMENTING CRM SOFTWARE**

---



---

**An e-book by Richard Boardman from  
Mareeba CRM Consulting**

Version 0.2.- June 2010

# CONTENTS

Chapter 1: An Introduction.....	3
Chapter 2: Planning.....	6
Chapter 3: Requirements gathering.....	12
Chapter 4: Vendor selection.....	16
Chapter 5: Design.....	25
Chapter 6: Development.....	28
Chapter 7: Data.....	30
Chapter 8: User acceptance testing.....	33
Chapter 9: Training.....	35
Chapter 10: User adoption.....	37
Chapter 11: In closing.....	40
About the author.....	42

## Chapter 1

---

# AN INTRODUCTION

---





## An Industry Insider's Survival Guide to Everything You Should Know About BUYING AND IMPLEMENTING CRM SOFTWARE

I wanted to write this e-book because there's frighteningly little practical advice for people who are new to buying and implementing CRM technology. The content that is out there is invariably written by marketing people from CRM technology companies, or by journalists writing what they've been told by the marketing people from the CRM technology companies.

And since most marketing people don't spend a lot of time at the front-line trying to implement CRM technology, and even if they did I dare say they wouldn't tell you too much about it, because there's a little more complexity than meets the eye, and complexity can be a dangerous thing when you are trying to sell lots of software.

So, this guide is a little bit different in a number of respects. It's written by me, a poacher turned gamekeeper, who left the industry to become an independent CRM consultant six years ago, and gives an inside perspective on a lot of things the industry would probably prefer you didn't know.

It's also a practical survival guide, based on 15 years of front-line experience and involvement in over 300 CRM implementations, of all shapes and sizes, to give you the information needed to purchase the right CRM software, at the right price, and help you pick your way through the pitfalls of the implementation process, to, hopefully, smoothly deliver a system that really makes a difference to your organisation.

The topics covered should be relevant regardless of size or type of CRM implementation, and the approach as applicable to a 'software as a service' (where the software is remotely hosted - often referred to as 'in the cloud') or on-premise deployment (where the software is run on your own servers).

The guide is broken into a number of chapters that follow the general flow of the purchase and implementation process, as follows:

- Planning
- Requirements gathering
- Vendor selection
- Design
- Development
- Data preparation
- User acceptance testing
- Training
- User adoption



## An Industry Insider's Survival Guide to Everything You Should Know About BUYING AND IMPLEMENTING CRM SOFTWARE

If you are a reader of my blog—[The CRM Consultant](#)—some of the content may seem familiar. I've borrowed from some of my previous posts, and edited and combined them with new material, to hopefully provide a reasonably coherent and logical flow.

I'm intending issue 'new and revised' versions of the book based on reader feedback in the future. So, if you have any input or insight based on what I've written, please don't hesitate to let me know (at [feedback@mareeba.co.uk](mailto:feedback@mareeba.co.uk)). You can also sign up [here](#), if you would like to receive new versions of this guide, as well as others we have in the pipeline on related CRM topics.

Finally I've designed this as a genuine self-help guide. The objective is to equip you with the information you need to successfully buy and implement CRM technology. While my firm (and others) can provide independent CRM advice and consultancy, this document has been written on the basis that you will 'fly solo' on this project. We're there of course if you need us, but this guide is not designed as an advert for our services.

Anyway, I wish you all the best with your project, and I hope you find this book of use. If you have any questions, feel free to log them [here](#), and we will do our best to provide a timely response.

**Richard Boardman**

June 20<sup>th</sup> 2010

## Chapter 2

---

# PLANNING

---





## Why are we doing this?

While it might not sound like the world's most exciting activity, planning is the cornerstone of a successful project, and as such gives a very high return for a modest investment of time. At its heart is the development of the business case. In other words defining what problems you are looking to solve or desirable outcomes you are looking to achieve.

There are several reasons clarity here is important:

- CRM technology is a tool. It won't produce value on its own. It needs to be used in a coordinated way to produce results, and there are many and varied ways in which CRM technology may benefit your business. Therefore, without a clear objective to guide your project from the outset, it's unlikely to generate value.
- Unless you can convey the benefits of the project in a compelling way, it's unlikely you will secure the necessary financial investment or, perhaps more importantly, the necessary injection of internal attention and resources required for success.

As you start to build the business case, it's worth noting that there are two broad ways that CRM technology may improve the operation of your business:

- **Process automation**—where you take what you do currently and improve things through better supporting technology. For example, you might have excellent existing processes in terms of how you attract, develop, and retain customers, but these may be supported through a range of Excel spreadsheets, standalone systems, and databases. CRM technology might create new efficiencies by replacing these silos of information with a central system which allows customer information to be better shared and more beneficially used. In this case your underlying business processes may be adapted to CRM technology, but they are not fundamentally changed.
- **Process development**—where the business processes themselves are re-engineered, or entirely new processes are created. For example, you might adopt a different strategy in terms of how you manage sales leads, or streamline the order management process, or change the way you handle customer issues and complaints. In this case, existing processes may change radically, and CRM technology plays a key role in their successful adoption by the business.

Most CRM implementations tend to focus on process automation simply because it's generally easier to replicate what you do already. While process automation projects can produce a high pay-back, the greater returns on investment are generally achieved through the process development



## An Industry Insider's Survival Guide to Everything You Should Know About BUYING AND IMPLEMENTING CRM SOFTWARE

approach. The organisations that use CRM technology the most effectively, in my experience, tend to focus on achieving process 'best practice' and use systems like CRM to drive those best practices through the business.

With either approach, the starting point is to analyse and document how business processes are currently performed, and how they are supported by technology. If the objective is to automate, rather than introduce new process, then the analysis should focus on identifying efficiencies that can be gained through better automation. This does require a working knowledge of CRM technology that you may not currently have. However, as many CRM technologies are available to evaluate free of charge, and, as the general concepts and capabilities of different products are similar, it is not an unduly time-consuming task to gain the necessary knowledge by reviewing some of the mainstream packages.

The simple act of documenting existing business processes often produces a few surprises in terms of how things are *actually* done as opposed to how it was believed they were done. This may in turn move a project away from process automation to process development. It should be noted though that re-engineering existing processes and adding new best practices is a more challenging and time consuming activity than simply automating what you do already. There's no single way to go about doing this, and it can be the product of careful analysis, internal brainstorming, consulting with customers, researching how other top-performing companies operate, and accessing the knowledge of external consultants.

However they are achieved, the output from these exercises should be some clear statements regarding the beneficial outcomes. For example:

'By streamlining and automating the order process, we expect to reduce the time to fulfil orders by two weeks, and reduce the cost of processing them by 40%', or:

'By better identifying and segmenting our customers, we can better create relevant communications and offers with a view to increasing our sales to the customer base by 20%'.



## Avoiding the big bang

As part of the planning process thought should also be given to how the project is phased. Doing too much in one go is a recipe for failure. The following are a few key things to bear in mind when considering phasing:

- **Do the minimal amount that gets results**—it's easy to over-engineer CRM systems, so it's generally better to implement something reasonably simple which generates quick results, and then build on it. This approach reduces the risk of spending a lot of time and money creating capabilities that later prove to be white elephants.
- **The first phase must be value generating**—whatever you do in phase one must create compelling value. If it doesn't, you will struggle to get resources for later phases. I see a lot of vendors promoting the 'suck it and see' approach, where customers are encouraged to buy some CRM software and then 'sort of experiment'. This might be good for short term software revenues, but rarely produces systems that clients want to continue to invest in.
- **Resources dictate phasing**—getting people to use a system in a consistent and structured fashion is one of the key challenges of CRM deployment. User adoption requires people on the ground winning hearts and minds, and this tends to be resource hungry. Therefore one of the key determinants of phasing is the amount of available resources. Try and do too much in one go and the implementation team can quickly become overwhelmed.
- **User micro-phasing to maximize adoption**—there's only so much change that users can embrace at one time, so breaking down a single phase into a series of smaller releases over the period of, say several months, can be an effective way of addressing the user adoption bottle-neck.
- **Schedule subsequent phases in advance**—if your CRM project is to be phased, it generally makes sense to ensure that the timing, content, benefits, and costs of future phases are broadly defined up front. This helps ensure resources are available when you need them, and avoids the need to go through a lengthy capital allocation exercise for each subsequent phase.
- **Reporting must be phase one**—for reasons that I explained [here](#) previously, CRM vendors seem to discourage users from worrying too much about reporting in the early phases of a project. Since reports are the key way for the management team to ensure that the processes that the system supports are being followed, relegating them to the 'manyana' file virtually guarantees system obsolescence.



## How much will it really cost?

Once you're clear on the objectives and the phasing, the feasibility of the project should be considered by estimating the costs of realising them. Vendor quotations can be fairly readily obtained, but these are unlikely to be more than ball-park estimates at this stage, and, because vendors tend to underestimate (they generally don't want to scare potential buyers off), they should be treated with some caution. There are also a lot of costs that generally won't appear in these quotations, but which also need to be considered. These potentially include:

- Server hardware costs
- Database software costs
- IT infrastructure changes such as required upgrades to hardware, software, and networks
- Project team costs—the additional real or opportunity costs associated with people involved in the project being taken away from their 'day-jobs'
- Third party project resources
- Ongoing system administration costs

It's also wise to validate costs by speaking to people who have been involved in similar projects, and also to include a high degree of additional contingency—perhaps 50 to 100% more costs than your initial estimates suggest, is probably a reasonable rule of thumb.



## Is everyone on board?

Assuming the project makes commercial sense, then the final step is to ensure the potential project has strong backing from all levels of the organization. There is a fundamental point here: if the management team is not prepared to use and manage the business through the CRM system, CRM will not produce any meaningful value.

I will make the point again (and underline it for additional effect) because it's so critical:

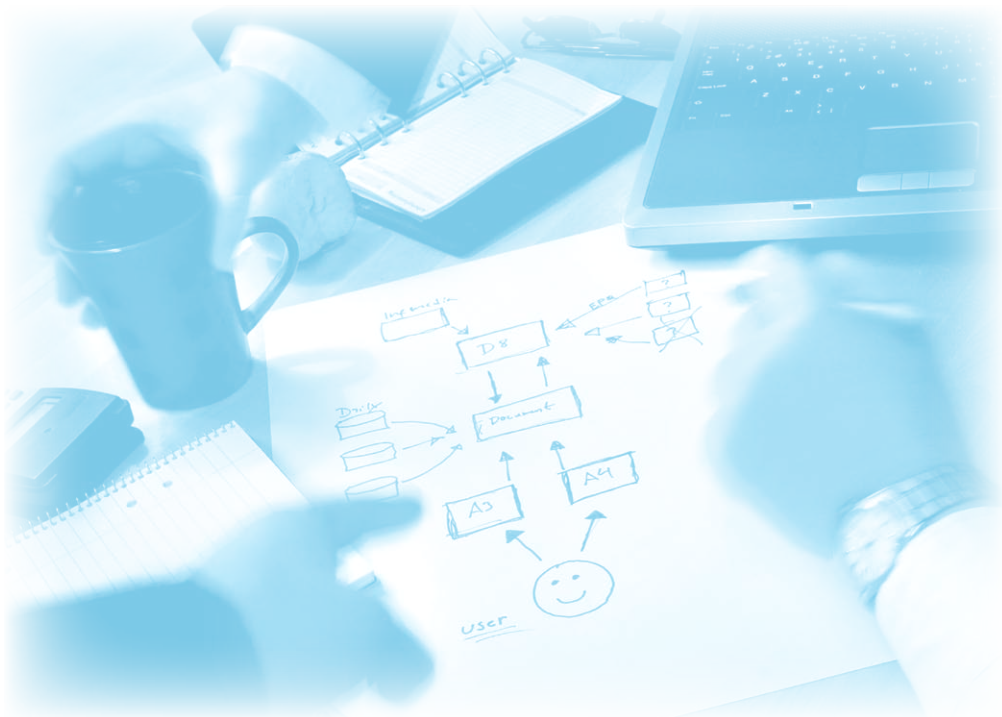
**If the management team is not prepared to use and manage the business through the CRM system, CRM will not produce any meaningful value.**

So even if the business case is strong, and there's an obvious return on investment, if the management team are not committed to it, then it's better either to wait until the planets are more favourably aligned, or look at undertaking the project in a different way. This might involve finding a smaller part of the organisation, perhaps a team, function, subsidiary, or office, who fully buy into your ideas, with whom you could prove the concept, and build support for a wider deployment in due course.

You will note that I haven't talked about buying any software at this stage. A lot of organisations rush to do this and regret it later. If you are happy that the plan makes sense, and you have the required support for it, the next step is to define the requirements for the project in considerably more detail.

## Chapter 3

# REQUIREMENTS GATHERING





## Why's it so important?

Getting requirements defined correctly is probably the single biggest determinant of CRM success, for the following reasons:

- It helps ensure the project receives the funding, resources, and management attention required to succeed because there is clarity about how the system will benefit the organisation. A lot of CRM projects fail to be adequately resourced because no sufficiently compelling outcomes and means of achieving them are defined.
- It reduces the risk of purchasing an inappropriate technology because the detailed functional requirements are understood before the technology is selected rather than after.
- It allows organisations to reduce costs, because vendors can provide firm quotations in a competitive environment against your detailed specification. In contrast, where requirements are poorly defined, the best a vendor can provide is an estimate to be confirmed at a later point. The 'later point' is a poor position to negotiate from, because by that stage you are generally heavily committed to the supplier.
- It speeds up delivery of the project, because a detailed specification means that the implementation phase is quicker.
- It improves cash flow, because the requirements gathering phase—one of the most time-consuming parts of the project—is completed before you start spending money on software.
- It reduces the risk of cost and time overruns, because there is less likelihood of new requirements appearing as the implementation progresses (often referred to as scope-creep), and because there is less likelihood of your selected vendor 'discovering' more complexity as they understand your needs better.
- It gives you more flexibility in managing the project, because it's easier to reprioritise work should the need arise.
- It accelerates adoption, because users better understand why they are being asked to use the system. Users tend to adopt technology considerably better if they can identify desirable outcomes if the system is a success.
- It increases the return on investment because there is a clearly defined business objective and means of achieving it.



If you can get the requirements gathering stage right, then this makes everything else that follows considerably easier. Unfortunately, people tend to struggle with this phase, and this generates a lot of unnecessary issues downstream.

## The mystery of the missing process

One of the most common issues with the requirements documents I get to see on my travels is that most are just bullet-point lists of required functionality. There isn't any mention of the processes that the technology will support. This is a potential problem for a number of reasons:

- It makes it very difficult for prospective suppliers to accurately assess the cost of customising the system because there's insufficient detail for them to do so
- It can cause lengthy delays during the implementation stage while processes are finalised
- If the processes aren't finalised, then effective user adoption will be impacted because there's no common understanding of how the system is to be used.
- It increases the risk of selecting a technology that's inappropriate to the purchasers needs because the detailed needs aren't fully understood.

It's also something of a tell-tail sign that the prospective purchaser hasn't fully understood that CRM technology is a tool and will not generate any value unless it's used to achieve well defined objectives.

The starting point is to begin with the desired outcomes for the project and then work backwards to identify the processes required to achieve those objectives. For example, if the goal was to increase the frequency and quality of communications to prospects and customers, then it would be appropriate to consider processes such as initial data capture, marketing campaign tracking and management, data segmentation, maintaining and improving data quality, managing opt outs and 'gone aways', lead and enquiry tracking, response and return on investment tracking, reporting, and the synchronising of data changes with other systems.

As you look at each of these processes in detail, and work out how you would want them to be supported, you will get a much clearer understanding of what fields you will need to track in the system, the associated functional needs, as well as the initial data migration and ongoing integration requirements.

In essence, a process oriented approach to requirements definition will help you flush out the detailed functional needs, which in turn will allow you to get a firmer fix on costs, better identify suitable technologies, as well as provide a route map to reach your end destination with less risk of expensive detours.



## Documenting your requirements

In terms of structuring your requirements document, I've found the following approach seems to work well:

1. An overview of the current situation, the problems you are looking to solve, and a statement of the desired outcomes. This should be as specific as possible.
2. A detailed definition of the business processes necessary to achieve the objectives, and how these will be supported in the system. I tend to break these into two parts: a narrative describing each process, and a flow-chart representation which includes a commentary as to who is updating what in order to facilitate the process.
3. The supporting functional requirements. I tend to start with a detailed description of each entity (for example people, organisation, lead, opportunity, and case records) within the system. This will include a detailed breakdown of what fields will appear on each entity and any related functionality. It's also worth adding mock up screen shots which can be quickly created using something like Microsoft Visio, as this visual depiction allows people to more easily review the document. Any remaining functional requirements that don't relate directly to an entity, such as the often overlooked areas of reporting, administration, and security, should be listed under separate headings
4. The details of all data integrations to, and migrations from, other systems. There's a tendency, in the CRM requirements specifications I see, towards broad-bush statements such as 'the system will integrate into system x' with little information about what 'system x' is or what information is to be integrated, in which direction, or in what form (for example in real-time, overnight batch loads, or as a data view). It's virtually impossible for a prospective vendor to gauge the complexity and cost of integration unless you can provide the supporting detail.

The resulting output should be a substantial and comprehensive document that will facilitate effective technology and vendor selection, and drive the implementation process forward in a controlled way.

## Chapter 4

---

# VENDOR SELECTION

---





As I set out in the previous chapter, the key to selecting the right CRM software is to have a detailed set of requirements. Only having high levels requirements—or none at all—makes it difficult to distinguish between the various offerings, because the functional needs are unlikely to be fully known, and pricing proposals will be indicative estimates which may prove to be very different from what you end up paying.

Having detailed your requirements, the first step in the vendor selection process is to identify a good initial list of potential technologies and suppliers.

## The start list

Not surprisingly, you're likely to make better choices if you are selecting from a group of top-flight technologies and suppliers. Conversely, if you start off with a list which is largely inappropriate for what you are looking to achieve, then you simply end up selecting the best of a bad bunch.

You may note that I mention both CRM technologies and suppliers. Generally—though there are exceptions—CRM software is sold or implemented through a network of independent resellers or implementation partners, rather than the company that developed it. The selection of the reseller or partner is at least as important to success as the choice of technology itself. It's therefore important to think in terms of these two dimensions when researching the market. And, as the ability and professionalism of resellers and implementers varies very significantly, this should be an area where you take particular care.

In this respect I'd be wary about recommendations from the software vendors about which resellers they think you should use. A lot of buyers will treat these recommendations as gospel, however they are invariably based on factors that suit the software vendor rather than the purchaser, for example, if the reseller is likely to make a quick sale (rather than implement the software well), or if you happen to sit in a reseller's designated 'territory'.

I generally look to get six to eight suppliers lined up to respond to a request for proposal (RFP). This gives a little leeway in case a few fail to respond. We also work hard to reassure the vendors that it's an open contest (which it always is). If vendors harbour any suspicions that the decision might already be made and that the purchaser is just going through the motions, then they are unlikely to bid. Good vendors are generally very busy vendors, and responses to RFP's are time consuming, so it's worthwhile making the effort to promote the opportunity to them. This may sound a little counter intuitive, after all surely they should be doing the selling, but the value of working with a great vendor over a mediocre one, massively outweighs the effort.



## The request for proposal

The next step is to prepare and distribute a request for proposal (RFP) document in order to better evaluate the available options. While some purchasers are inclined to skip this stage and start directly auditioning potential suppliers, getting written submissions from vendors gives you the opportunity to compare and contrast the offerings in a much more structured and analytical way, and gives you a better basis for action should disputes over what was promised arise in the future.

The goal with the RFP document is to strike the balance between getting the information you need to make a decision, while making it as easy as possible for the vendor to respond. If you make the RFP too onerous then there is a risk of potential suppliers deciding not to bid. As I mentioned before, a good vendor is likely to be a busy vendor, and they will weigh carefully the potential for successfully winning the business against the time and cost involved in preparing the response. So it's best to keep the latter element as light as possible.

The RFP document should set out how and when the vendor should respond, and then prompt the would-be supplier to answer questions on the following areas:

- The profile of the software and its developer.
- The profile of the software vendor/implementer, particularly in relation to support capabilities, implementation approach, and experience with similar projects.
- Cost structure, including a detailed breakdown of both vendor supplied elements, such as software, services, training, and support and maintenance, as well as any other relevant costs such as hardware and database software.
- The final element is a table referencing the detailed requirements document, asking potential vendors to explain how they deliver each identified requirement, and provide man-day estimates for any that will require customisation/development to deliver.

While some of the vendors we work with may choose to disagree, we find this approach is light enough not to discourage responses, while providing us with sufficient information to make informed judgments about suitable candidates for the next stage: creating the short-list.



## The short-list

Having received the responses, the next step is to review them and reduce them down to an appropriate short-list. I tend to grade them based on the following six questions:

1. How good is the functional fit?
2. How does the purchase price compare?
3. Can this vendor deliver a quality implementation?
4. Does it appear they want this project?
5. Are they going to be easy to work with?
6. Will they still be a strong supplier in five years time?

It's worth noting that the information supplied to you shouldn't be unquestioningly relied upon. There will be a lot of half-truths in tender responses along the following lines:

**RFP question**—'Does your software perform function xyz'.

**Response from vendor**—'Yes',

**The truthful answer**—'Yes, though it involves using a third party module the price of which I haven't quoted, it will also involve a considerable amount of customization, and it only works with the enterprise version of the software (and I'm quoting you the entry level version) and actually, while the module does do what you want, it's not the best designed piece of software and your users will soon abandon it because it is unfathomably complex to use.'

So, if you have any suspicions that you may not have been given the whole truth, particularly if it's in relation to a key capability, don't hesitate to question it carefully with the prospective vendor.



## The demonstration stage

Having determined a suitable short-list of prospective vendors—I've generally found four candidates to be a reasonable number—the next stage is to arrange for the vendors to present their products and credentials, normally in the form of a software demonstration and presentation.

This demonstration phase should be treated with a little caution. I know well from my time working for vendors how easy it is to script a demonstration in a way that showcases the offering's strengths and skates over key weaknesses.

Software demonstrations, rather like job interviews, are a somewhat flawed process. A candidate may be great for sixty minutes in an interview, but it's no guarantee they will perform over the long term. The demonstration produces its own distortions; I've seen excellent products from highly capable vendors culled from candidate lists for seemingly trivial reasons, and products I knew couldn't do the job, from vendors who I wouldn't trust to mow the lawn, end up winning the day.

So, while demonstrations are important, the polish of the presentation or the charisma of the salesperson should not unduly influence the decision making process. In fact over the years I've often found that there is an inverse relationship between the slickness of the sales process and the quality of final delivery.

To counter the presenter's 'slight of hand', I've found it useful to inject as much structure as possible into the presentation/demonstration process, in terms of what to cover, and how long to cover it for. I'm also a great fan of giving prospective suppliers specific demonstration scenarios which reflect how the system will be used in due course. This makes it a lot easier to gauge how well the software fits the processes it will potentially support in real life.

Also, given that the successful salesperson is likely to disappear to the Caribbean for a 'well earned' holiday when the real work needs to be done, it's a good idea to meet the people who will be involved in the implementation, particularly the project manager. You will be dealing with this team, not the salesperson, on a day to day basis, so it's important to check they are capable and, perhaps more importantly, people you feel you can work with over the duration of the project.

At the conclusion of the short-list presentation stage you ideally want to be in the position where you feel comfortable that you could work with any of the vendors. This gives you more options moving forward. Hopefully at this point though you can identify a preferred potential supplier, though if it's close this could be two, and move forward to the next stage.



## An Industry Insider's Survival Guide to Everything You Should Know About BUYING AND IMPLEMENTING CRM SOFTWARE

It's important though that the preferred supplier isn't allowed to become too confident about their chances of being successful, because this can significantly impact your negotiating position in due course. It's also not that unusual to reach a dead end with a 'preferred' supplier and need to revisit the original short-list again. Therefore it's important that all the candidates understand that it remains an open contest.

The next step is to verify whether the preferred supplier is indeed the right choice. As part of this, I like to visit the vendor and get a better feel for how they run their business. This is also the opportunity to discuss any outstanding queries, meet other project team members, and discuss any potentially ticklish contract terms. It's also a good idea to establish contact with relevant members of the supplier's executive team, as you may need these lines of communication later if issues crop up on the project.

Many people will follow this up with reference calls, though this should be treated with caution. Even bad vendors will be able to rustle up a few positive references. However, as most suppliers like to showcase their customers during their sales presentations, a more insightful alternative is to take notes and make your own enquiries without going through 'official' channels. This is likely to give you a much truer picture as to a vendor's reputation.

Assuming things progress satisfactorily, you now need to finalise the price for the project. In general it's better to get the vendor to commit to a firm price, rather have them working on a time and material basis where they are effectively incentivised to take as long as they can.

If your requirements have been defined in detail as previously described, this should be a fairly brief process. The vendor should be able to review the requirements, and, with a relatively small amount of discussion and clarification, arrive at a fixed price. If you haven't been able to get to a suitable level of detail, then more work is likely to be involved. The prospective vendor will need to undertake their own requirements analysis work which will generally be a chargeable exercise.

The key though is to only pay for the requirements work at this stage, and not to commit to the project as a whole. If a vendor is working on finalising a price for the project, and you are already committed to them because you've already bought the software for example, then you have very little leverage if they decide to take advantage of the situation and ramp up their estimates.

In general I tend to be very uneasy if there's any significant difference between the price provided in the original RFP response and that provided at the end of this process. Unless there's a reasonable explanation, I may elect to back track a little and conduct a similar process with other suppliers, which, again, is why it makes sense to keep your options as open as possible during the vendor selection phase.



## Negotiation

As a final part of the process you may wish to negotiate pricing and terms. While it's in your interests to structure a project so the vendor can make money on it—otherwise they may try and cut corners later—trimming out any unnecessary fat can make a big difference to the final purchase price.

While this may involve negotiating down software prices or day rates for implementation work, it's the amount of implementation work that you need to be careful of because it's easy for vendors to pad this out if they wish to. So, if the vendor is quoting ten days to perform development work that should only take two days, you are going to be significantly overpaying, even if you've successfully negotiated down the day rate.

When you are looking to negotiate pricing, here are a few tips:

- **Look at timing**—vendors are much more likely to offer discounts at key points in their financial year, such as quarter and year ends.
- **Don't show your hand**—the tendency to discount will reflect the degree of confidence the vendor has of securing the deal. If they are confident, they tend to discount less than if they feel there is a realistic chance they may lose the opportunity.
- **Have options**—you are in a much better position to negotiate if you have three or four viable suppliers, than if you have just one.
- **Research**—a little knowledge about discounts a vendor has offered in the past can go a long way to help secure a satisfactory conclusion.
- **Get outside help**—if you are unsure if the number of days you are being quoted for development work is fair, get independent advice from someone familiar with working with your preferred technology.
- **What can you offer?**—vendors are much more inclined to negotiate the greater they see the value of you as a customer. Could you be a reference for them? Might the system extend to other parts of your business? Is your decision likely to persuade other companies to buy? Is this a new market or a new application of their product? If you can communicate your full value as a customer, the keener the vendor is likely to be to secure your business.
- **Don't get pressurised**—vendors love to make pricing concessions based on you ordering by a certain date i.e. 10% off if you order by 31<sup>st</sup> December. These are generally artificial devices designed to force a quick decision. A quick decision may not be in your interests however, so progress things at your own pace, the discounts will almost certainly still be there when you are ready to move forward.



## Contractual terms

Getting the right price is important, but meaningless if the contractual terms are not right as well. Be aware that anything involving contracts has the capacity to take a very long time, and, if you are working to tight time-lines, the earlier you can begin the process the better. So, it's worth making sure that both parties have visibility of any contentious terms during the RFP stage.

With respect to negotiating terms, I'm not intending to offer any specific legal advice, but the following are few areas you may wish to be mindful of when finalising an agreement:

- **What does the software agreement allow you to do?**—It's important to be clear as to how many users are allowed to access which capabilities of the system. For example, most vendors have different versions of their software from say entry level to enterprise, and it's important to understand which version you are licensed to use and what the associated restrictions are. It's also worth checking whether you have the right to run a separate instance of the software for training or testing purposes.
- **Are all the costs known?**—there can be a lot of hidden costs when purchasing CRM technology. Additional storage costs can be an example in a hosted environment, or maintenance agreements that are free of charge when you buy the software but kick in heavily in future years, for on premise software. Make sure these costs are fully disclosed in the agreement, and it can also be worth negotiating a cap on a vendor's ability to raise prices to unreasonable levels in future years.
- **What are the payment milestones?**—I generally prefer contract payments to be built around the achievement of specific project delivery milestones, rather than the vendor invoicing on a weekly or monthly basis. This helps focus vendors on delivery rather than billing. In terms of software, when do the licenses need to be purchased? Vendors like to invoice on order, but if the implementation time-lines are likely to be lengthy and no one is using them before they go live, then this can unnecessarily tie up capital. Equally support and maintenance agreements should ideally start from 'go-live' rather from the order date.
- **What's the basis for payment?**—In general I prefer fixed price contracts where the costs are known in advance and there's less scope for cost overruns, rather than time and materials contracts where there's no incentive to the vendor to complete the project in a timely fashion.
- **Who owns the intellectual property?**—By default a supplier will own the intellectual property rights to any customisation or development work they perform on your system.



## An Industry Insider's Survival Guide to Everything You Should Know About BUYING AND IMPLEMENTING CRM SOFTWARE

If you subsequently fall out with them, you may find that you no longer have the right to use the software, therefore it's important that the contract addresses this ownership issue. There are a number of options which range from you owning the copyright, which the vendor may be reluctant to grant, to having a perpetual license to use it. In working this through you may also want to consider how you would feel if your chosen vendor offered the customisation work they had performed for you to your closest competitor!

- **How can an agreement be ended?**—Whether the agreement is for support services, or a software as a service (SAAS) contract, it's important to be clear how the agreement can be terminated. Ideally you are looking to have as much flexibility as you can, whereas it's generally in the vendor's interests to restrict you as much as possible. I particularly dislike contracts that are automatically extended unless the vendor is notified by a certain date. If a vendor is unduly focused on locking you into a contract, it's generally a red flag in terms of their confidence in their ability to deliver a quality service.

### Vendor selection—a summary

The combination of a comprehensive set of CRM requirements, and a well structured approach to the vendor selection process, helps ensure that you choose the most appropriate technology for your needs, and purchase it at a fair price. I estimate that it's on average 30–40% cheaper to purchase technology with the 'front-loaded' requirements led approach set out above, than the more traditional approach where poorly defined requirements mean prospective vendors are unable to provide firm pricing. This approach also speeds up the implementation phase and reduces the risk of later overruns.

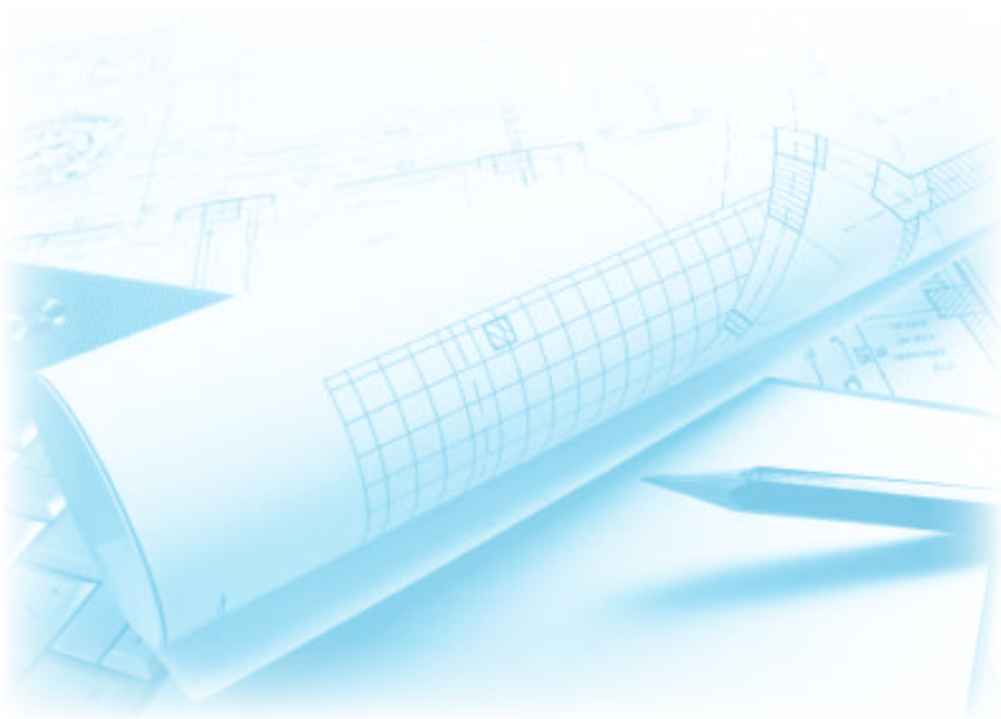
Anyway, now we've selected our vendor, it's time to implement the system.

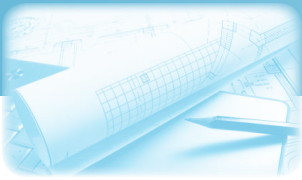
## Chapter 5

---

# DESIGN

---





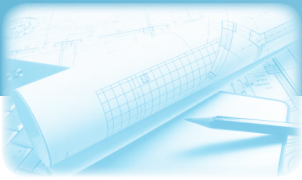
## An Industry Insider's Survival Guide to Everything You Should Know About BUYING AND IMPLEMENTING CRM SOFTWARE

**S**o having selected our CRM vendor, and having negotiated satisfactory terms for the project, it's time to start the implementation work. The first activity is normally a design phase where the vendor works out how your requirements can be best accommodated within their technology.

On the surface this might seem a relatively innocuous piece of work, but it's one that trips up a lot of organisations. The thing to be aware of is that what you agree at this stage is what the vendor will go off and create, and if what they create doesn't turn out to be quite what you want, then the vendor is likely to point out that, since you signed off the design specification, they'll now have to start all over again—and of course charge you for the privilege—so the project ends up taking a lot longer and costing a lot more than you intended.

Reviewing design documentation though can prove challenging, so my six tips for surviving the experience:

- **Start with a good set of requirements**—Yes, I've mentioned this several times so far, but suffice to say if you have a detailed well thought out set of CRM requirements—apart from saving you lots of money—it gives a ready means of checking off that everything you previously said you wanted is indeed what your vendor is planning to give you (vendors having tendency to slightly selective memories when it comes to things they feel might be awkward/expensive to develop).
- **Do not expect it to be in a language you understand**—Design specifications are written for two audiences: you, as the client to sign off, and the vendor development team so they know what they're meant to be developing, but actually mostly the vendor development team, who are generally extremely technical, which you may or may not be, and work with the selected software day in day out, which you probably don't. So it's a bit like being asked to sign a contract which is written in, say Greek, when English is your first language. So allow a lot of time—even a straightforward design document can take several days of work to fully review.
- **But make sure you understand every word**—Even innocuous phrases can have deep meaning, so these documents don't suit a quick skim. The greatest mistakes I've made on projects have been where I've figured something's been too arcane for me to take the time to fully understand, so my advice is no matter how dumb you feel the questions may be, keep asking them until you are 100% certain you understand what's being said.
- **Expect trouble**—You might be wholly convinced that your selected vendor are a pretty switched on bunch, and they might well be, but they will still make mistakes and omissions, and a lot of them, so don't get lulled into thinking that you can trust them, and avoid



## An Industry Insider's Survival Guide to Everything You Should Know About BUYING AND IMPLEMENTING CRM SOFTWARE

the necessary investment in the time to fully review the document. Even on a modest requirement I figure the vendor has done a pretty good job if I find less than a hundred issues on the first pass.

- **Finished reviewing?—the journey just begins**—While you might feel a sense of a job well done having completed your review of the design, there's normally plenty more to do. Curiously, many design specifications have more mistakes in their second iteration than the first – a phenomenon for which I'm unable to offer any logical explanation. Half a dozen iterations isn't that unusual for a moderate amount of customization, so budget accordingly.
- **Make sure the vendors have the resources available to action your input**—You'd figure it wouldn't surprise vendors that clients actually read their documentation, but it seems to. Once you've completed your review you would hope the vendor would be able to process your input in a timely manner, but if the relevant staff member has been allocated onto another project this isn't going to happen, so it's always wise to make sure that resource is available to quickly process the necessary changes, otherwise very lengthy project delays can ensue.

The design phase might seem innocuous, but it really does catch a lot of organizations out. Very substantial overruns can easily result, so it's really worth the considerable investment of time and energy required to get it right. There aren't unfortunately any short-cuts, but I've personally found drinking a lot of coffee seems to help.

## Chapter 6

---

# DEVELOPMENT

---





## An Industry Insider's Survival Guide to Everything You Should Know About BUYING AND IMPLEMENTING CRM SOFTWARE

**I**n general, the development phase is the where the end user of the system is least involved. This is often a fairly brief interlude in the project where you can focus on getting ready for taking the system live. The key word here though is brief, which surprises a lot of people, as there's often a perception that this is a particularly time consuming part of the project. In reality, with the flexibility of a lot of the available CRM packages, this stage can be completed relatively quickly and actually only occupies a small percentage of the time for the overall project.

This assumes however that you don't fall victim to something called 'scope creep'. Scope creep is generally the result of flaws in the requirements gathering process, and means that new needs are added to the specification while development work is underway. This is a problem because it can result in considerable re-work which increases costs and delays delivery.

While there will nearly always be some need to tweak or enhance the system based on feedback from users, the more thorough and detailed the requirements gathering work (sorry to mention it again) the less chance that scope creep will significantly impact the project.

As a final point on development, as we will discuss in the next section, the quality of the development work will have a huge impact on the duration of the user acceptance testing stage. I've sat through many, many presentations over the years, which all made a point of the rigour of the vendor's internal testing procedures. In practice though, software often arrives full of very obvious issues for which the client bears the cost of finding and logging. It's worth therefore stressing to the supplier the importance of delivering a quality product at the test stage, and retaining the right to reject it if quality isn't at an acceptable level.

## Chapter 7

---

# DATA

---





## An Industry Insider's Survival Guide to Everything You Should Know About BUYING AND IMPLEMENTING CRM SOFTWARE

One thing that catches a lot of people out is data preparation. You may develop a great system, but if the data quality is poor when it goes live then users are unlikely to be impressed. The problem for many organisations is that data is spread across many systems and spreadsheets, with varying degrees of attention having been paid to maintaining data quality. The following problems typically apply:

- **Data is duplicated**—the same data resides in multiple systems. For example, the details of a key customer might appear in an old CRM system, the financial system, and ten Outlook contacts lists. When you bring all the data together, you will have many instances of the same contact, and these duplicates will be carried over to the new system unless you take steps to de-duplicate them.
- **Data is inaccurate**—the data source has been poorly maintained. Perhaps little attention was paid to inputting details accurately and there are many spelling mistakes and categorisation errors, or maybe the source data was difficult to interpret, for example someone was keying in customer data from a hand-written warranty card.
- **The data is out of date**—so you might have a spreadsheet with the attendees of a trade show you attended three years previously. On the surface this might look like useful data, but how many of those contacts have moved on, or their details have significantly changed?
- **Data is incomplete**—key data is missing. While the data might be both up to date and accurate, how complete is it for the purposes you wish to use it for? For example if you wanted to send a personalised letter just to customers who bought your Model 4C Widget Maker, last year, do you have all the data you need or will there be gaps? Perhaps I've only got the initial of the buyer and I don't know if they are a Mr, Mrs, Miss, or Ms. Or perhaps I have the contact data but don't know what machine they own.

The problem with data preparation for a CRM project is it's time-consuming. While there are tools that you can use to improve data quality, there's still a lot of manual intervention involved. So, a few quick tips:

- **Start early—don't leave it too late.** Data preparation can take many months, so the earlier you start the better, ideally as soon as you know the project will happen.
- **Be ruthless**—do you really need every data source that's been identified? Will the cost and time involved in improving the quality of some of the sources outweigh the benefit of bringing in the data? The complexity of bringing data into a new system is multiplied by the number of sources, so cutting back here can make a big difference to implementation costs and time-lines.



## An Industry Insider's Survival Guide to Everything You Should Know About BUYING AND IMPLEMENTING CRM SOFTWARE

- **Treat data as a key part of the project**—it's easy to see data preparation as a minor element in the CRM implementation plan, but in reality it can have a big impact on the success of the project. Make sure that data preparation is fully catered for and resourced in your project planning.
- **Get outside help if necessary**—CRM projects are time-consuming, so if it looks as if data preparation work could overwhelm internal resources, then consider contracting in temporary staff or agencies to help.

## Chapter 8

# USER ACCEPTANCE TESTING

<i>Poor</i>	<input type="checkbox"/>
<i>Satisfactory</i>	<input type="checkbox"/>
<i>Good</i>	<input type="checkbox"/>
<i>Excellent</i>	<input checked="" type="checkbox"/>



## An Industry Insider's Survival Guide to Everything You Should Know About BUYING AND IMPLEMENTING CRM SOFTWARE

Once the CRM vendor has completed their development work, the system will be passed to the client for user acceptance testing (UAT). The objective for this phase is to spot any potential bugs and validate the system is going to meet the needs of the users.

In principle, what the vendor should provide you with is a well tested system which can practically be waved through UAT. In practice however, their rigorous testing phase may get dispensed with if the vendor finds them self under time or budget pressure, and the client effectively gets landed with the vendor's testing responsibilities, or worse, gets to try and test something that's effectively still in development—the equivalent of putting up the wallpaper while the plasterer is still at work. What might have been a relatively trivial piece of work is transformed into a death march as the bug count ratchets up and up.

Just to make this all slightly more pressurized, because UAT is pretty much the final step before live, most of the associated live activities are all now scheduled, and there's often relatively little scope to move dates out to reflect the unexpected influx of work, and so the test team ends up absorbing the workload the best way they can—generally dispensing with life's luxuries such as sleep.

There are however a few things that can be done to help address this:

- Have a detailed, mutually agreed and understood design specification that gives you something to test against—this limits the scope for misunderstanding as to what should be delivered
- Make sure you have plenty of resources available to do the testing. It's a time consuming activity which should involve both members of the project team as well as key users. If people don't have plenty of time allocated to this phase it won't get done properly.
- Pay for implementation work on the basis of achieving defined milestones, and make sure one of those milestones is the delivery of a high quality system into UAT.
- Make sure your vendor has made resources available to quickly fix the issues you identify, because if developers get allocated to other projects, you could be facing a long wait.
- Assume the worst, because you'll generally be proved right, and allow more time than you possibly think you'll need.

In summary, while UAT is one of the final hurdles in the implementation process, it's been responsible for more than its fair share of project delays, and has tempted many a project into the potentially fatal act of going live with a part-cooked system. As with many aspects of CRM implementation, it's worth treating cautiously.

## Chapter 9

# TRAINING





## An Industry Insider's Survival Guide to Everything You Should Know About BUYING AND IMPLEMENTING CRM SOFTWARE

With user acceptance hopefully now complete and with 'go-live' looming up, it's time to train users how to use the new system. The following are some of the key considerations:

- **Do a lot of it**—the amount of training required to help users change existing practices shouldn't be underestimated. Giving users half a day's training and expecting them to start using the system in consistent and structured fashion is a triumph of hope over reality.
- **Training should reflect process**—the training should be tailored to your specific processes and supporting customisations. Too often users are trained on the 'out of the box' software with little emphasis is how they should be using the software in context to their own organisation. This does not foster effective usage.
- **Training should be role specific**—the things you want your telemarketing team to do with the system may be very different from the pre-sales team. A one size fits all approach is going to dilute training effectiveness.
- **Don't forget the management team**—they may be very busy, but, as we've touched on before, if they don't use it, the project will fail, so it's important that they are fully trained during this phase.
- **Don't limit yourself to classroom training**—the classroom has its place but one to one training should be proactively targeted at individuals struggling to embrace the technology.
- **Look for trouble**—don't assume that if there's no screaming everything is alright. A user should not be considered trained until you've validated they are using it in a consistent and structured fashion.
- **Training is an ongoing programme not a one off event**—new staff will join the business and they will need to be trained, and that training needs to be done in a thorough, structured way, rather than a ten minute demonstration from a colleague in a rush to get on with their day job. In addition there will be a requirement for refresher training for existing users and further training when software is upgraded or new capabilities added. You should therefore budget for an ongoing training programme if you are going to get the most from the system.
- **Make the right decision between vendor and in-house training**—there isn't a simple answer to which is better; it will vary from organisation to organisation. In-house (i.e. you use your own staff) training is likely to be more cost effective, but is predicated on having someone that can communicate well, and the time and inclination to make it work. Vendor training is generally high quality, but tends to be expensive and many vendors struggle to customise their training approach for individual clients. Whichever choice you make, monitor the quality of what's delivered, nothing kills a system quicker than bad training.

## Chapter 10

---

# USER ADOPTION

---





## An Industry Insider's Survival Guide to Everything You Should Know About BUYING AND IMPLEMENTING CRM SOFTWARE

I was recently asked to help a software company brainstorm how they could get more value from their CRM system. They had clearly spent a lot of money on it. They had a full time administrator, the software ran on a bank of perfectly maintained servers, and they were a prominent reference site for the vendor whose software they used. After a few minutes looking at their system it was apparent they had a problem though. No one was actually using it. To be precise, out of a sales force of a hundred and twenty, only seven had even accessed the system in the previous month.

The moral of the story is that you can have the greatest CRM system in the world, but if you can't get people to use it in a consistent and structured way it won't generate any benefit for your organisation. User adoption has been the primary cause of death for most failed CRM initiatives in a world where there have been a *lot* of failed CRM initiatives.

In reality, user adoption is not that complex an issue to address, but like many aspects of the CRM world it's considered to be a technical issue. So, as each new CRM software product is released, the vendor, correctly identifying user adoption as a key point of failure, announces another breakthrough in ease of use capabilities that will supposedly banish this ill forever.

Putting aside my belief that actually there has been very little change in the usability of CRM software in the last 10 years, ease of use is only one piece in the user adoption puzzle, and it doesn't matter *how* easy to use you make software, it doesn't mean it *will* be used. The following are the bits that I believe make up the bigger picture:—

- **Have a clear purpose**—as we've already noted, it's important to define compelling objectives for a system, but it's also critical that users understand and buy into those objectives and it's not just seen as another pointless system they have to update. Communicating this effectively is the key here.
- **What's in it for me**—while it's important that the CRM system will create benefits for the overall organisation, it's critical that it also provides benefits to the individual users in terms of making their lives easier, or helping them work more effectively.
- **Don't compromise on ease of use**—it's generally worth spending a bit more on development work to make things more ergonomic or intuitive for the user, because these investments generally pay big dividends in improving uptake and usage.
- **Half a day is not enough**—let me be emphatic here; do not expect your users to attend a classroom training session and immediately start using the system in a consistent and structured way. This will not happen. Classroom training is a starting point, but as I touched on earlier, it is just part of a sequence of activities that will need to happen before you get strong usage.



## An Industry Insider's Survival Guide to Everything You Should Know About BUYING AND IMPLEMENTING CRM SOFTWARE

- **Assume failure**—in other words assume that people are not using the system in the way you wish them to until you prove otherwise. Have a list of system users and only cross them off when you've proven they are using it. Until then, keep training.
- **Resource up**—user adoption activities, as perhaps is becoming clear, can be time consuming. Make sure you have the people in place, whether they are internal or external, to ensure that you can get through this phase of the project.
- **Middle management is key**—in many cases the senior executive team will buy into a CRM project, but this buy-in doesn't extend to the middle management tier. Nothing will kill the system off quicker than a user's manager intimating that use of the CRM system is optional. Effort should be focused on ensuring that all levels of the management team are driving the use of the system.
- **Monitor**—there should be someone whose role it is to carefully monitor usage. If users are not following the agreed processes then that needs to be logged and addressed. This is a key function, so make sure the chosen staff, are capable, motivated, and have the time available to do the job properly.
- **Accept it takes time**—user adoption won't happen overnight, it will take months, maybe years. Get it right and it will be worth it, but accept this is a long term activity.
- **Burn the boats**—when the Spanish Conquistador Cortez landed in Mexico he ordered that the ships be burned to ensure his soldiers were solely focused on the task ahead, and knew there was no heading back. If the only way to perform a given task is through the CRM system—perhaps generate a sales forecast, or the monthly activity report, or place an order—then the system will be used. If there are other means to do this, perhaps manually, or through an old system, it reduces the motivation to change.
- **Reports**—are a key way for users to determine if processes are being followed. If they're not, then you will quickly know it because there won't get any usable management information. Strangely though, reports are often a belated afterthought encouraged by vendors who point to the standard reports that ship with the software, or wizzy report generators. In reality, since reports need to track your individual business processes, they often require fair amount of customisation and development, so make sure that this is factored into the schedule so they are ready to support the user adoption process.

## Chapter 11

---

# IN CLOSING

---





## An Industry Insider's Survival Guide to Everything You Should Know About BUYING AND IMPLEMENTING CRM SOFTWARE

I don't think any of the approaches I set out in the previous pages are unduly complex, but they often run counter to the prevailing market wisdom, which is perhaps why relatively few investments in CRM technology yield real dividends. However if you are prepared to take on board the following key points, then I believe you have every chance to be successful:

- Be clear about what you are looking to achieve
- Produce a detailed set of business and functional requirements before you start to select a vendor
- Take the time to carefully research potential technologies and especially implementation partners
- Don't rush it—successful projects take time
- Don't underestimate the time and resources required to get people to use a system in a structured and consistent way

Hopefully by following the advice in this e-book you've gone a long way to smoothly delivering a system that adds real value to your organisation. Please note however, delivering a successful project is just the start. CRM systems are fragile plants, and keeping them generating returns over the long term often isn't easy. People come and go, organisations can change dramatically in short periods of time, so keeping the system relevant and productive is not without its challenges.

Anyway, that's a little outside the scope of what I wanted to cover in this guide, and is perhaps a topic for another e-book. If you want to be the first to receive future publications though, you can sign up [here](#).

Well that's all from me. I wish you all the best with your project, and I'm always happy to try and answer questions if you have any. The simplest way of doing this is to use the Mareeba 'Ask a question' page which can be found [here](#). I look forward to hearing from you.

# ABOUT THE AUTHOR



Richard Boardman is the founder and principal consultant at independent CRM consultancy, Mareeba ([www.mareeba.co.uk](http://www.mareeba.co.uk)). At the time of writing Richard has worked in the CRM industry for 15 years and has been involved in over 300 CRM implementations. He's a poacher turned gamekeeper having worked for a number of CRM vendors before setting up Mareeba in 2004.

Richard is a regular commentator on the CRM industry and author of 'The CRM Consultant' blog. He can also be found as [@crmadvisor](https://twitter.com/crmadvisor) on Twitter.