

Daniel Shefer  
[www.shefer.net](http://www.shefer.net)

With input from Steve Johnson of [Pragmatic Marketing](#), Rob Steckbeck and Rick Chapman, author of [The Product Marketing Handbook](#) and [In Search of Stupidity](#).

## **Desktop vs. Enterprise Applications – The impact on Product Management**

A couple of months ago I discussed product management with a VP at a company that is moving from the “tool space” to the “enterprise application space”. As a result of that conversation, I asked myself, how is product management different for enterprise software vs. for desktop applications? This article is an answer to that question.

There are many differences between managing a desktop tool or application and managing an enterprise-ready software package. The impact of buying an enterprise application is more profound on the buying company, the sales process is much longer, and its participants and nature are different. Customer expectations of functionality, support, services and pricing are different as well. These differences impact the product management process, product features and capabilities and how Product Managers interface with Sales, Professional Services, Support and of course with customers. Consider the following:

- Customer investment: Enterprise systems are a significant investment in the order of hundreds of thousands of dollars into the multiple millions of dollars. Products of this magnitude carry risks that must be mitigated. Will the product be disruptive with existing business practices? Will you, the vendor, be in business tomorrow? How is the product supported? The more perceived risk in an enterprise purchase, the less likely it will happen.
- Customer value: Customers investing at an enterprise scale need to be confident in the product's value. What is the return on their investment (ROI)? What is the risk involved in obtaining this ROI?
- Complex environments: An enterprise product must scale and is touched by many types of users. The enterprise product must fit into the existing and possibly very complex enterprise application environment. Can the product be integrated into the enterprise and who undertakes that integration task?
- Complex sales: There are multiple people involved in the buying process and the product needs to address them all. Some call this “buying by committee”. The purchase is usually part of a budget set aside for this purpose. Approval of the purchase by a committee or a “higher authority” is a pre-requisite to placing an order. There is a lengthy period of time from the initial ‘interest’ until a decision to purchase is made. It's not uncommon for a complex sale to take 9-18 months. Sales cycles that address problems that are not in the top three on the customer's pain list, can suffer two sales cycles: one to convince the buyer of the magnitude of the problem, and then after a period of waiting for budget, another complete sales cycle to the new players. And then, the Product Manager is often brought in to provide a product roadmap as the voice of authority on the product's direction and future.
- Services: Service issues are far more critical in the enterprise sale. In fact, the service portion of the sale may be more important than the actual product sale.
- Post Sales: Post sales issues are more important in the enterprise sales cycle. When a new version of the software is available, customers usually can't just download

it and run an upgrade script. Issues that need to be considered are currently deployed versions, data to upgrade, backups and contingency plans in case something goes wrong.

This article discusses some of these issues from the perspective of the Product Manager. At the end of the article, an appendix details architectural and technical issues that need to be addressed to make an application “enterprise ready”.

## **The Product Management Process**

### **Requirements are Much More Complex and Lengthy**

When defining the requirements for an enterprise application, these tend to be much more complex and lengthy than the requirements for a desktop application. The number of interfaces with other applications, functional requirements, types of users, security and other enterprise related requirements entail much more complex requirements.

### **Getting the Voice of the Customer is Harder**

Getting good input to the requirements process is harder when managing an enterprise application. There are two reasons for this:

1. The end user who will be using the application is usually not the person buying the application and may not be present during the sales process.
2. The buyer has a different set of requirements and expectations than the end user and many times, these contradict each other.

### **Not All Development Processes Can be Used to Develop an Enterprise Application**

Agile and XP development processes are more relevant to the development of smaller, more “stand alone” applications. Agile is best when a customer can truly sit next to a developer and works better for internal IT projects. In enterprise applications, one cannot access a customer on a daily basis and the Product Manager serves as the voice of the customer. This means that Product Managers must truly understand the customer needs and environments. It also means that the Product Manager cannot be constantly out on sales calls, thus away from the developers when they need input. The nature of enterprise applications is such that a waterfall development process is used in more cases than not.

### **Coordination Becomes Much Harder**

By nature, enterprise applications are more complex than desktop applications. They have more internal modules and interfaces with other applications. These interfaces include billing and reporting applications, identity management services etc. The interfaces have to be designed-in at the beginning of the design process and managed throughout the lifecycle of the product. As the number of versions of both your application and the applications it interfaces increases, managing these interfaces becomes much more difficult.

## **The Product Marketing Process**

The go-to-market planning and execution on the product marketing side of things for enterprise applications is many times under the control of the Product Manager. When selling desktop applications via channels, the latter tend to drive much of the exposure to and interaction with the customer.

## Who are you selling to?

The sales process of enterprise applications is fundamentally different than that of desktop applications. Desktop applications by their very nature are point solutions with a single-need buyer. Enterprise products are sold either to the IT Department or the IT Department is involved in the buying decision. The sales cycle is inherently more complex and longer. Enterprise products are usually sold through a direct sales force rather than through channels. The product marketing efforts are significantly different when supporting a direct sales force of an enterprise application.

When making a complex sale, the sales rep will meet multiple people in the buying organization. These can be the user, the decision maker, the IT department etc. Strategic selling defines three roles in the complex, consultative sale:

- economic buyer: the one who says "yes"
- technical reviewer(s): the ones who are interested in the nuts and bolts and try to say "no"
- users: the ones who implement and use the product (they are typically involved in the evaluation but not always)

## Sales Materials

When creating sales materials, multiple audiences must be taken into consideration. There are the buyers, the techies who make the recommendation to buy and of course the end users. Each of these has different expectations and may require a separate set of marketing materials.

- Economic buyer: Expects an easy to achieve Return on Investment (ROI) and a financial justification for the purchase. Product Managers need to provide proof of ROI for their product as well as detailed Total Cost of Ownership (TCO) for widely deployed applications. All claims need to be solid as buyers tend to be skeptical and references need to be presented.
- Technical reviewer(s): Expect technical specifications, whitepapers, explanations of how the system works and comparisons with your competitors.
- Users: Expect a clear picture on how the product will make their work easier and more productive. Users need to be able to get hands-on experience of how the system will benefit them in the form of trail software or a convincing demo.

## Sales Support

When selling desktop applications, most products are sold via distribution channels. Sales support is mostly about helping the channels create demand by driving product positioning, advertising and supporting their sales efforts with onsite sales collateral, canned sales pitches etc. The people selling the product will be limited to rudimentary sales and product training (if even that).

With enterprise sales, the size of the deal is significantly larger than desktop applications and there is a lot more time and resources investment from the sales reps in the process. Because of this, sales reps will try their best to drag the Product Manager into the sale whenever they feel it will give them an advantage. Product Managers will be asked to talk about the product, present the roadmap (a document that should never be given to reps), and even make commitments to customers (which they should try to avoid).

Another difference lies in the nature of the sales materials. High end and specialized collaterals, such as vertical whitepapers, case studies and competitive materials will also have to be developed to support a major push into a market.

## **Product Management and Professional Services**

Vendors selling desktop applications rarely offer professional services to their customers. Someone buying a \$500 desktop application would probably not consider a service to support or enhance the software that would cost many times the original amount.

Enterprise applications need to be able to offer basic functionality out of the box just like desktop applications. However, customization is many times needed to tailor the application to the client's business processes. Professional Services' responsibility is implementation of the product, change management and how the product impacts the customer. Think of Professional Services as the "adoption machine" of your product. You can have a good product that will die without professional services and a lousy product that will get installed everywhere if it has the right support.

Professional services have to be designed as a key element of the offering to make this work.

### **Professional Services in the Planning Stage**

If the product requires professional services to get its full value, these services need to be planned in when designing the product's architecture and not as an afterthought. For example, an API that drives the business logic layer needs to be created at the "get go" and documented for the Professional Services group to make their offering meaningful.

### **Professional Services as Your Secret Ally**

The Professional Services team is more the voice of the customer than the Sales team. Sales reps can describe what the "buyer" wants, but rarely what the users really do with the product after the purchase. Product Managers need to work closely with the Professional Services team during the requirements collection stage. Their input is critical if 40-60% of revenue is to come from existing and happy customers whether they upgrade or buy more licenses.

## **Product Management and Post Sales**

### **Rolling Out Upgrades**

Rolling out an enterprise application to an existing customer poses different challenges than rolling it out to a new one. In the former, issues such as upgrading the existing data is a major concern and having a roll back option are major concerns. Doing so by setting up a parallel environment is extremely difficult. When rolling out a system to a new customer, in an ideal scenario, the disruption is minimal and limited to the systems that the new application will be integrated with. Careful planning and execution can avoid disruptions.

### **Training**

When enterprise applications require training for their successful deployment, it is Product Management's role to provide the various internal groups with the information to create the training materials they need. This can come in the form of documents such as a "what's new?", a document that describes how the new functionality should be explained and how it benefits customers and fits into their "world".

The Professional Services group needs input and/or training to create training materials for customers. Additional audiences for specialized training are the Sales Engineers and Support reps. Sales Engineers are critical to the complex enterprise sale, if only because the sales person needs more technical credibility. Both groups need extensive technical,

hands-on training and it falls on Product Management and Engineering to provide this. This training has to begin before the first Alpha release and progress as the product matures towards customer availability.

### **Customer Testimonials**

Testimonials from successful deployments of an enterprise application are an important part of the sales tools. Enterprise customers want to hear testimonials of other companies that solved problems similar to their own. Many large companies have strict processes and procedures about giving testimonials. Some completely prohibit this. It is either up to Product Management or MARCOM to contact the customer and drive the process of getting a testimonial written and approved.

### **Summary**

In this article I highlighted some of the differences between managing an enterprise application vs. managing a desktop application. As shown, these two types of applications require slightly different product management skills.

In the appendix, you will find a more detailed list of architectural differences between the two.

Good Luck!

This article and its contents copyright (c) 2004 by Daniel Shefer.

---

---

## **Appendix – Technical and Architectural Aspects of Enterprise Applications**

The following is a list of functionality that enterprise applications must address to be considered “enterprise ready”. For a more detailed discussion of the topics below, see Luke Hohmann’s excellent book: [Beyond Software Architecture](#).

### **Security**

Police may tell us that they are here to “serve and protect”. IT Departments see their duty to “protect first and only then to serve”. Security is a “Sine Qua Non” requirement – without it, you cannot proceed in the sales cycle. As a vendor, you can never invest too much in the security of the application. Enterprise products should be designed from the get go to pass a security audit and certification process. The following are some of the issues that need to be addressed.

#### **Identity and Rights Management**

System Administrators of enterprise applications need to know who has access and to what part of the application. This is required if they are to limit users to specific parts of the application and track changes to the system. Changes to the system need to be limited by the user’s rights and logged.

Furthermore, user accounts have to be managed. Doing so manually for a few users is easy. Managing thousands of accounts is not a scalable proposition and has to be done hands free integration with a directory server such as Active Directory or iPlanet so the process of managing accounts is automated.

## Transaction Security and Traceability

A significant problem with enterprise application is guaranteeing the integrity of the data. For example, what happens to a transaction if the user gets disconnected from the network? After a transaction is completed, it needs to be logged. If the system has accounting or other legal implications, log files need to be saved for years.

Another aspect of transaction security is the risk of eavesdropping. If the application's data goes over the Internet, it must be in encrypted form. The standard today is 128 bit SSL encryption. Some customers may ask for stronger encryption such as AES (Internet Explorer does not support this yet). In any case, note that encryption technology has export restrictions so when selling localized versions to foreign countries, you may have to offer less than the maximum encryption available for that version.

## Application Security

If the application is installed internally, it has to withstand reasonable hacking attempts such as blocking access to a web page when a user did not go through the required authentication procedure. Log-in failures have to be logged, minimum password requirements enforced etc.

If the application is accessible on the Internet, the above requirements will be more stringent and you may be required to have the security of the application assessed by a third party. Furthermore, customers will expect protection from DOS attacks (a networking issue), certificates for file integrity verification, anti-virus scanning, C3 level hardening of the hardware etc.

## Data Integrity

Measures have to be taken to prevent and identify any changes to the application's files and data. Some of the data such as passwords and personal information needs to be encrypted and access to them must be limited to specific administrators.

## Web Based Applications

### Browsers as clients

A robust, dedicated client is always better than a browser based client. Isn't Outlook better than Hotmail? If the software client needs to support significant functionality, a user, a dedicated "fat" client makes the most sense. If instead, the client is used casually by a large number of people, a web client is ideal. Browser based clients have an inherent advantage as they do not need to be upgraded on the client side.

Browsers have inherently less functionality than an installed client. The way to get around this is to use a Java applet that resides in the browser. IT departments are more inclined to allow Java based applications in a browser than full blown installations but don't be surprised if a prospect has their Java Machine shut off as the default IT policy.

Regardless of the decision to use a dedicated versus a browser-based client, your sales team will need a strong argument explaining your choice. If the decision was largely a technical one, a white paper is the best way to address a potential objection.

### Bandwidth

If the application accesses the Internet, the question of how much bandwidth it requires will come up in the sales cycle. If there are multiple users accessing the Internet simultaneously, significant bandwidth usage may be an issue. An inside the firewall caching solution or support for one will come in handy in such a case.

## Network Access - Firewalls and Proxy Servers

Network Administrators can be very imaginative when it comes to security policies. Expect to find network configurations such as proxy servers that require authentication (NTLM etc.), firewalls that limit outgoing connections to port 80 and 443 and even some that disconnect sessions after a set time. One company I'm familiar with has internal browsing done on one port and Internet browsing on another. Another security policy is NAT (Network Address Translation). It is almost standard so if the application needs to know the originating IP of a client inside the network, your developers are in for a bit of work...

If your application has to access the Internet, this is a tough one. If your application requires an open inbound port in the firewall, forget it. You will not be able to demo the application from behind the prospect's firewall without a dial out connection. The application must be able to determine independently how to access the Internet. This can be done in several ways, the most common is by detecting the local PC's network settings from the browser or locating the networks rules file and parsing it. As the way these files are written is not standard, this approach will require additional work and constant tweaking on Development's behalf.

## Supported Platforms

Which operating systems an enterprise application needs to support is a more complex issue than with desktop software. On the one hand, supporting more platforms makes the sale easier. Customers can choose whatever combination of operating system and version they like; one less cause for an objection. On the other hand, supporting more platforms than is absolutely necessary is a very costly proposition that can critically drain Development and QA resources.

When deciding which operating systems a desktop application will support, all one has to do is look at the list of Windows versions that Microsoft currently supports. Linux and other UNIX derivatives are currently superfluous for the desktop market. Re. Macintosh, it's up to the sales forecast to determine that.

On the enterprise side, there are significantly more operating systems, web servers and databases available and each one has several versions deployed. Enterprise customers do not run to upgrade their systems when a new version comes out. The number of potential combinations can be staggering.

Another issue is when to support new versions of customer systems. Few customers will expect you to support the latest and greatest if it's new or several months old. Even if the product requires a stand-alone installation and your customer is willing to buy new hardware and software for it, they may have to be a version older than the newest available as customers prefer versions that have been found stable and whose problems are known.

Another factor is the extent of existing customer investment. If a customer is a Solaris "shop", they may be reluctant to buy an application that runs on another operating system. Luckily, the religiousness of IT departments on this issue seems to be declining. The bottom line is that it is worth while to make an effort to support as few combinations of operating systems as possible but keeping it within reason.

A side note: If you think that your product supports too many platforms, an extreme case is [Perforce](#). They support no less than 114 combinations of operating systems and software versions. Ouch!

## **Networking Issues**

### **IP Addresses**

Is the product web based? Does it require a static IP? If so, in many cases, to install it at a customer site, you will have to go through a lengthy process to obtain one. With one company I worked with, getting a static IP took three weeks. Keep the number of IP addresses required by the server to a minimum.

Some networks, for reasons of security and web content filtering, do a reverse DNS look up. This means that when the browser looks for the IP address of your hosted server, the proxy server looks up the server name to make sure that it is not a restricted site. If your product uses IP addressing instead of domain names, this can result in poor network performance.

### **Installing Client Side Software**

In many corporations, end users are not administrators on their PCs. In some cases, users cannot install anything and in others they can only install applications that do not write to certain places in the Windows Registry. The only way to distribute an installed client application within such a customer is with a network management suite such as SMS or OpenView. IT may also expect you to provide an MSI installation script to assist the process.

One thing that has to be addressed when designing the installation of the client side software is the issue of personalized applications. For example, if your application relies on user specific data to run the first time, how this information gets sent out to end users is an issue that needs to be addressed.

In many companies, there is a pre-defined and rigid hard drive structure. For example, the C:\ drive is reserved for applications and the user's content is saved on a network drive. The installation must support this and offer administrators flexibility in choosing where to install the product.

Another common situation is where installations are not allowed to update system DLLs. If the installed application does require updated system DLLs, make sure that there is a simple way to have these placed in the application's own folder. In any case, if System DLLs are updated, make an effort not to require a reboot of the PC after installation. System Administrators and end users hate rebooting.

Some IT departments have the OS configured so it will not allow any Registry keys, content nor any application files to remain after uninstalling. Make sure that the product's uninstall is squeaky clean.

### **Localization**

In many cases, a pre requisite to selling the application abroad, is localizing the user interface and translating the documentation. Together, these can easily cost you \$30,000-\$100,000 or more for each version, depending on the complexity of the application and your development process.

### **To ASP or not To ASP**

Making a decision about offering desktop applications in an ASP model is easy as there doesn't seem to be a significant market for this (McAfee may disagree). For enterprise applications, the decision is much more complex. On the one hand, ASP served applications are easier to get up and running, require less set-up costs from the customer

and are easier for you to support and maintain. On the flip side, this model has three fundamental flaws.

1. The software is not behind the customer's firewall and therefore will be perceived as less secure.
2. When an application is hosted, it cannot be integrated with back-end systems due to security concerns (the need to access sensitive applications that are behind the customer's firewall). Many benefits of an enterprise deployment are thus lost.
3. The customer's data is held captive by the vendor. For whatever reason, individuals are comfortable with using an ASP model for buying and selling stocks but extremely uncomfortable with putting their checkbooks anywhere but on their computer. How would a CIO like to put all their customer's information in your hands? And if your company fails, where is the data?

## Deployment

### Training

It's acceptable that training be required for a small number of specific users such as in the case of an ERP system. An enterprise wide deployment of a product that requires training for all may be a difficult proposition to for customers to accept.

### The Roll Out

If the application has a complex product you will need a roll out plan. The plan should describe the steps needed to roll out the application, timelines, who does what, critical paths, milestones, potential pitfalls, success criteria, best practices etc. If a big consulting firms has their signature at the bottom of the roll out plan, all the better.

### Updates and Patches

A major issue with enterprise roll outs is the version update policy. Rolling out the application can be an arduous process, taking many months. Coming out with new versions every 3 – 6 months and a couple of patches in the middle might solve some of your issues but from the customer's perspective, it's very disruptive. That's one reason large companies run old versions of applications.

Quarterly releases still make sense, but not for general availability. Instead, put it on the shelf until a customer requests it. New sales will always get the latest version while existing customers will only get what they need. At most, an enterprise application is expected to have one major release each year. Major releases might have changes to the GUI, the data structures, major new functionality etc.

If a customer complains of a bug that was solved in your latest version don't be surprised if they demand a patch for their installed version and not want to upgrade to the new one. From their perspective a new version means new, unknown bugs, uncertainty and a resource intensive upgrade process that they will have to deal with. Why bother? Customers want the problem solved but without any hassle. It's a good argument so you'd better be prepared. This is one of the many reasons that every new version should have a "sexy" feature set to entice people to upgrade.

The distribution of patches and their frequency requires careful planning. Using Java and browser based components has a big advantage when the software is constantly updated. An auto update feature is very helpful when the client software is installed.

However, it has to abide by the IT Department's rules and they must be able to fully control the upgrade process. Remember that any upgrade process has to be scalable. When upgrading an enterprise application, setting up a parallel environment to test the new version is many times impossible. Therefore, it's critical to plan an upgrade path while keeping existing systems in place.

### **Upgrading Data**

A critical problem with upgrading existing installations is the data that has been created. While simple in theory, in practice the data schema may change between versions and a detailed mapping of how the data changes must be created and communicated to the customer before they approve the upgrade.

### **Backups**

The product has to be flexible as there are so many techniques and products for backing up computers and databases. Some companies prefer to back up the whole computer, OS and all and others will only back up the database. While straight forward, the optimal backup technique needs to be documented up front.

Also, many companies require that accounting and other business-related data be archived for 3 – 7 years after creation. The software needs to support this type of data preservation from "day one".

### **Performance**

If the application is a client server type, IT departments will expect a detailed description of the resources needed to run it. You will need to detail how many transactions are supported per second per CPU, RAM and relevant network configurations. Customers also expect a clear upgrade path to the next performance level. For example, will it require an additional CPU or more RAM or can another server simply be added to increase performance?

### **Integration**

Customers expect enterprise applications to integrate with their currently installed applications. Some things should come out of the box such as LDAP integration with Active Directory or iPlanet and with other applications with a well documented, standard based API. The API needs to be designed in when the system architecture is defined and not as an afterthought. As a rule, the business logic should be separate from the core of the application.

Having an API toolset no longer cuts it. Stating "We can support LDAP with our XML toolset" is a fundamentally different statement than: "customer X is using our out of the box LDAP support". IT buyers want to see working implementations that did not require custom code.

### **Pricing**

One positive aspect to having a direct sales process is the flexibility you have to choose a pricing model. The flip side is that enterprise buyers expect you to accommodate their specific pricing model preferences. This expectation can create difficulties as the pricing model of the software impacts the product's architecture. Change the pricing model and changes have to be made to the software that are not necessarily easy. So until your company is big enough to say to a prospect "no, we don't support this model", the

application's licensing mechanism needs to be flexible enough to accommodate at least a few pricing models.

Another issue is how the license is verified. Doing so on the server side is the hardest way technically. For example, how do you measure concurrent users? What happens if a user disconnects in the middle of the session? Does the server wait and for how long until it frees the access to another client? On the flip side, a product with a client that requires a registration key is easy to implement and may work well for small installations. However, for an enterprise-wide deployment, a method must be available to support an automated, hands free registration.

Another aspect of enterprise software pricing is the need to incorporate maintenance and service components into the pricing model. In many enterprise sales, these provide for the major part of income over time.

This article and its contents copyright (c) 2004 by [Daniel Shefer](http://www.shefer.net) - [www.shefer.net](http://www.shefer.net).

Additional articles by the author:

- [A Product Manager's Reading Anthology](#)
- [Disruptive Customer Demands](#)
- [Ten Things Product Managers Need to Know About Sales](#)
- [Nine Things Product Managers Need to Know About Supporting Sales](#)
- [Shortening the Sales Cycle](#)
- [Product and Pricing Strategies](#)
- [Webcasts as a Lead Generation Tool](#)
- [Creating Effective Competitive Sales Tools For Your Sales Reps](#)
- [The RFI as a Measurement of Product Marketing and Sales Reps Effectiveness](#)
- [Online Customer Forums](#)
- [Demos pilots and the sales cycle](#)