SAAS BUSINESS MODELS: KEYS TO SUCCESS

By Rich Mironov
INTRODUCTION

The purpose of this ebook is to provide software executives with a framework to understand how Software-as-a-Service (SaaS) business models work and how they can be successful in transitioning from an on-premise to a cloud-based deployment model.

Why Should I Read This?

Over the past decade, there’s been a massive shift in software deployment models from on-premise to the cloud. We will discuss why this shift has occurred, as well as best practices for how to approach a SaaS offering in terms of marketing, sales, and operations.

We will cover:

• On-premise vs. SaaS packaging and pricing models and how to maximize revenue
• Why a SaaS model makes you responsible for the customer experience from beginning to end
• What skills your organization needs to retain customers in a SaaS model
THE RISE OF SAAS

Each year, Logi Analytics surveys software providers to get a deeper understanding of what’s happening in the market and the transition to the cloud.

In the 2014 State of Embedded Analytics report, we found that cloud is now the predominant delivery model. 36% of software providers are deploying solely in the cloud and another 36% have a combined cloud and on-premise model.

Software Applications are Predominantly Deployed SaaS or Have a SaaS Option

36% of software applications are SaaS, and another 36% are both SaaS and On-Premise.

What’s more is that 52% of software providers say transitioning to a SaaS model is an important product management priority for them over the next 12 months.

52% of software providers say transitioning to a SaaS model is an important priority.

With the rise of SaaS, it’s important to discuss how deployment models affect delivery, pricing, marketing and user adoption, product development, and overall success metrics. We will discuss each of these in turn.
SERVICE MODEL THINKING: ON-PREMISE VS. SAAS

Remember the old days when cloud computing used to be called timeshare?

Watching several generations of product move forward is exciting, especially this latest major migration to the cloud. Most people who have come up through the product management ranks have been trained to think about products (particularly software products) as packaged, on-premise items where the important event is when a customer agrees to take shipment of the media. Back when we used to put things on tapes or cards or CD-ROMs, the big revenue event was actually putting it in the mail or handing it to a customer. As delivery transitioned to over the cloud, not only does the location of where the software lives change, but we’re also required to change how we think about the business model, how we think about marketing, and operations, and significant other parts of our product organization.
On-Premise Thinking

For on-premise software, revenue happens when you ship the box. To illustrate the point, let’s pretend your software is actually a box of cereal at the grocery store. With cereal, the important thing is how many boxes you shipped to the store, and you don’t care too much that people are actually eating it or how they might consume it once they get it home. You only really care about the metrics up until the boxes are sitting on the shelf at the store. Consumers are able to pick out different foods and ingredients that they assemble at home to make their own meals. Similarly with on-premise software, either the end consumer or the internal IT organization is responsible for picking the right things, putting them together, maintaining them with updates and patches and fixes, and deciding when they’re available for the business to use. It’s the responsibility of the consumer to create a successful solution here, not the software manufacturer, just as it’s up to the individual to whip up a tasty meal at home, not General Mills.

1 Actually this is where we originally got the concept of shelf ware because we had a lot of software that was shipped but never used.
SaaS Thinking

With an on-premise delivery model, software vendors are one step removed from happy users because they’re handing it off to someone else along the way. With a SaaS model, instead of a grocery store, it’s more like a restaurant. Restaurants lay out plates and silverware, prepare the food, and cultivate an ambience to help guests enjoy their meal. In the same way, SaaS vendors are responsible for delivering working software and guiding their users through a successful and pleasant experience. They’re in charge of availability, determining who gets access to what, and consumers can interact with them directly to get help and ask questions. The game just became a lot longer. The success of a SaaS model depends on repeat buyers and upsell opportunities along the way, and there’s no longer a single transactional moment that defines success. On the other hand, a single bad experience with your SaaS product is similar to an unhappy guest at a restaurant – it can turn them off of your business forever, and they can warn other potential customers away from you as well.
SaaS is the Default Delivery Model

We know the market drivers for SaaS on the consumer side: mobile business, heavier reliance on laptops vs. desktops, need for information at any time inside and outside the firewall. But there are also factors on the supplier side. I’ve seen it countless times over the last decade that when software companies get out of the business of hosting their own software it’s because consumers are demanding it and because it’s not very efficient or cost-effective for the company. Hosting demands lots of talents that software companies don’t need to maintain anymore, and from a software development point of view, it’s at least 10 times less expensive to build your software to be delivered in the cloud than in some mixture of on-premise and cloud or just on-premise.

Between managing the different versions and getting customers to upgrade and all the inter-operability testing etc., the current generation of software is almost exclusively being built in the cloud, for the cloud, and on the cloud.

Marc Andreesen, American entrepreneur, investor, and software engineer (most famous for coauthoring Mosaic, the first widely used web browser), is famous for saying “software is eating the world.” We are continuing to see that of course there are some situations where software has to live on-premise or in a particular piece of hardware for strategic or technical reasons. For example, if you develop routers in which case the software you build to power your routers lives in those routers or other hardware. Certainly if you’re in the printer business or in the up-and-coming 3D printer business, your software value lives in a printer, so clearly you’re going to ship it that way. I’ve also worked with financial or military application companies whose specialty is security and they often insist on managing and running their own servers or data centers. And of course, if you have a very high performance application such that you’ve got to do a very high throughput to an adjacent device, then your software might need to live as close to that adjacent device as possible. For instance, in the real time ad serving world there are some very high performance requirements. But in general, the default model for almost every class of software is in the cloud.
SaaS at All Software Layers

This change in mentality is happening in all software layers, where everyone is in the game of some kind of “as a service.” Certainly all the way down at the bottom companies like Rackspace are providing bare metal systems and basic operating systems as a service up in the cloud. Moving up through various kinds of platforms such as “AWS” and “Heroku,” there are all kinds of developer level support as a service. The point is, if you’re in the software business, you’re probably doing something as a service and there is a fair amount of positioning to do with your marketing organization to figure out just what you’re going to put in that first letter of position.
BUSINESS MODELS AND SOFTWARE PRICING

Having established that, let’s talk a little bit about pricing models because software as a service doesn’t by itself define how you’re going to exchange money or find value or demonstrate to your customers that you’ve delivered economic value.

The most obvious business model, however, is the subscription model, and implied in subscription is a unit of either months or years generally. For example, music streaming services like Spotify are licensed per month and online survey engines like SurveyMonkey tend to be per month or per year with an upsell. Data storage, CRM, domain registration, etc. also tend to follow subscription models. The benefits of a subscription model are straightforward – subscriptions roll over, they’re easy to buy, and they’re easy to sell. But they’re not the only model.

You may also decide to pursue a transaction model, where the important part of the transaction is to figure out what the unit is after the word “per” so whether you’re charging per click or per download, per stock trade, per movie ticket, per tax return. Software sold in this way tends to be a transactional service where the user may or may not come back very often or it’s a large single transaction and it just makes more sense to do transactional work.
The freemium model is also very well-known. In the freemium model, you assume that perhaps 99% or higher of all users will take your product for free but that you’ll get some upsell, some higher value, or extended feature which is going to get the 1% of users to actually pay off money. This model became popular in the early days by Google and the assumption that computing power continues to fall so rapidly that you can give away product and you can give away service to the vast majority of users at very, very low cost.

In the last 3-5 years, we’ve seen another trend where companies are offering not just end-user access through browsers or apps but they’re opening up their data directly through APIs as well. For example, if you look at Twitter, you’ll see that a majority of the input and output comes from third party applications. Similarly, anything to do with supply chain software or analytics software will likely offer APIs to their customers and partners instead of just end-user apps. You’ll need to treat APIs similar to core application to decide the best pricing model depending on whether they’re used transactionally, by subscription, or another basis.
SaaS Pricing Tiers

Once you’ve determined your pricing model, you’ll have to figure out how to set the price levels, or tiers. To illustrate this concept we’ll move from cereal to clothing analogies. When setting pricing tiers, it’s important to create a short, straightforward list of options and you want to avoid giving discounts or running “specials” as much as possible.

In the on-premise world, suppliers often get pushed into specials where they cut a different code line or product version for some of their largest customers and then maintain that long term as well as integrate it back at some point so they could get those customers back into the mainstream. You want to avoid this at all costs. Even if you’re offering a premium product or a trial product, the basic thing has to work. It has to be impressive; it has to coax users along because the sales cycle here tends to be basic to premium, not the other way around, so you need to be able to satisfy most of those users with basic products. Even those you may not be able to derive money from those versions, you can’t just have crippled or demo versions of things out there. It’s critically important that product managers, marketers, and developers alike understand what features are going to drive a customer from small to medium and medium to large. Again, the vast majority of users don’t need the advanced features but you do need to know who the folks are that do and why they need it in order to effectively upsell.
**Service Tiers/Bundles**

If you’ve been to any consumer-facing SaaS site, this chart probably looks familiar to you. I’ve genericized it here but you can see the basic product with some features and some support, then the expanded product represented by different levels, or tiers.

<table>
<thead>
<tr>
<th>Core feature</th>
<th>BASIC</th>
<th>EXPANDED</th>
<th>ADVANCED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core feature #1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Core feature #2</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Core feature #3</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Baseline support</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Baseline capacity</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Monthly newsletter</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Major upgrade feature</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Cool feature</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Capacity update</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Support upgrade</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Professional feature</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Tons of capacity</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Concierge service</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>CEO welcome letter</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>
In each step up, there is a major feature upgrade which is of course the item that you know this next audience needs and then some additional things that give it some bulk to make it even more compelling, like increased capacity and support upgrades. Then finally there’s the advanced version with professional features and maybe a letter from the CEO. It should be obvious why you’d want to follow a similar structure with your own offering: you have multiple opportunities to upsell the same customer, it’s easy to package, it’s easy for consumers to understand what features they’re going to get, and – from a software management point of view – it’s much easier to manage permissions, access, and various kinds of functional testing because there’s a limited number of combinations for what people are entitled to do. Any time you divert from this structure, however, it’s going to create a lot of overhead on the software and testing management side because every time we make some changes to our main code line we’re going to have to think about this one exception case.
Markets Define the Tiers

In any software company, developers are going to have strong opinions about product direction, but it’s important to remember that the optimal pricing tiers and feature bundles are determined by the market, so product management and/or marketing needs to act as the voice of the customer when defining your packaging. Again, the entry product needs to really work so web visitors are compelled to try it and users are immediately impressed by functionality and ease of use.

They need to be IMPRESSED before they’ll consider upgrading to a more premium version; they aren’t, they’ll move on to the next product.

Consider building in a couple cool features into the trial or basic level to generate buzz and interest to immediately show value to the customer. Think about how many seconds between when you signed up for Netflix account and when you were able to watch your first video – the goal for Netflix is to minimize the number of steps and the amount of time to value so the customer gets happy and starts exploring what else they could be doing in your product. For the product managers, it’s about looking into the different cross-sections of your user base to match use cases to features and creating an upsell path that makes sense and is compelling.

It all needs to be simple. The long-winded discussion from the on-premise days of which modules customers got and what features they got and how to turn things on must be transformed into a few bullet points on a webpage. As a SaaS provider you need your customers to be empowered to make a choice – the right choice – and be able to immediately get up and running on their own. It also has implications for overly complicated pricing or business models. For example, you might have “by the drink” pricing – for example if you charged customers every time they did a stock trade or every time they sent a fax through an eFax service. Or you might employ “all you can eat” or “all you can eat with tiers” or “volume tiers.” The important thing is you want to be in
one model or the other. It gets very difficult if you let all of your small customers buy “by the drink” and all your large customers by the “buffet plan,” and more than likely you won’t maximize your revenue if you take this route. It’s also more confusing for your customers as they’re trying to figure out what it is they’re trying to buy. If the pricing model is too complicated, customers will simply opt out entirely. Not to mention the fact that even if they opt in it will be difficult for you to report on what’s going on and track actual usage!

We’ve talked about attention spans briefly already, but it bears reminder that even considered purchases are getting shorter in terms of attention. Anyone who has watched their teenager download 20 or 30 or 40 new apps to their iPhone and cycle through them one at a time knows there is about 15-30 seconds for that teenager to get some gratification or to delete it and move on. On your own website you may only have 30 seconds to visitors’ value and explain your value proposition. Along with that we’ve seen a real migration uptick as all the consumer apps style a UI and good design work has finally filtered through the B2B products. Buyer expectations SaaS products are much, much higher than ever before in terms of UI, workflow, and uptime/bugginess. If something doesn’t work or doesn’t look right, consumers will immediately flock to the next thing. Therefore, as product managers and marketers for SaaS providers, the question you need to answer is how can you best demonstrate value to your users in the shortest amount of time? Can you do one quarter of a full search and return to them two interesting data items that will cause them to move forward? Can you retrieve one data record, find one interesting movie, and dredge one small piece of data out of your larger service in 20 seconds instead of setting up a 5 minute demo that nobody will complete? Think small and continuously make improvements until your users experience the aha moment.
THE AHA MOMENT!

Whenever you talk about showing value very quickly, a lot of folks talk about the “aha” moment, which ideally should happen within 30-45 seconds.

If you’re working on exchange rates, how can you fill the page first with a default exchange rate from the country of the person who is visiting as a good guess for what they need rather than making them click through six different options to get to the end result? Let’s think about CRM systems. I spend a lot of time with agile project management systems such as Jira, Rally, Version One, Mingle, etc. The way enterprises test out this type of software is they send one lead user into one account for one small subscription and try it out for a few days. If it doesn’t work, or isn’t enticing enough, you may only get 3-4 minutes worth of usage from that one person and without noticing you’ve lost the entire enterprise sale. Of course you’ll probably have lots of individual users of your product, but you won’t be able to tell which ones are solo practitioners and which ones are representing a thousand users in the back.

Remember, if you’ve made the transition to SaaS, you’re no longer in a one-time transactional sales model like you were with an on-premise product. The old style enterprise sales reps are turning into customer success managers to ensure that people kicking tires in your trial are having a great experience. When we poll SaaS customers, being able to kick around your software in real life and try it out, whether they have to pay for it or not, is the #1 benefit of a SaaS model to them.
What you end up finding is that the early pioneers are really part of a longer enterprise sales cycle where one or two or five people may get a subscription and then recommend it into the larger buyers.

We’re seeing that a lot in the agile project management space where most of the vendors have made THE FIRST 10 USERS FREE so I can go to any of these agile project management systems and sign up five or eight users, WHICH REPRESENTS AN ENTIRE DEVELOPMENT TEAM AT EFFECTIVELY ZERO COST.

If the first taste isn’t great, then they won’t be back. Think of your trial as an appetizer plate; you need to give them a good taste of what you’re about to give them so they have a reason to continue on.

This is even more critical from a revenue point of view, because you are pushing revenue out. You give the customer value first, and then collect on the value delivered later. With on-premise software, you put the software in a box, you ship it, and you took all of the money up front. Of course you recognize the revenue over 12 months or so, but all the money comes in at the beginning so there’s no risk until it’s time to upgrade. With a SaaS product, you are much more likely to see a slow ramp where more users are added over time and users upgrade to more premium versions over time, so it’s much more important that we’re making those customers happy now.
GETTING REAL FEEDBACK

Beyond efficiency gains, one major benefit of SaaS vs. on-premise is getting powerful, real-time feedback from your customers on the quality of your product.

In the on-premise days, software companies had to rely on surveys and user groups to figure out what they were actually doing with the product. Many of those methods are expensive, time-consuming, and can be biased to the extremes (either very happy or very unhappy customers are more likely to participate than people that think your product is OK). They’re also events, so the feedback takes awhile to get back to the product management and development teams. Depending on who you talked to, you might also question their motives – e.g. with sales we know their motive is to get that one feature for a deal to close this quarter.

In a SaaS model, you can actually have the data of what people are doing in your application. Of course you should still talk to people directly but having real-time, quantitative data is a huge advantage over the old model because you can actually determine which features are being used, which campaigns are working, what are the most common click streams, where are people getting stuck. You need to be very careful to protect your customers’ security and make sure you’re not intruding on them unnecessarily but this kind of information is a gold mine compared to one-off surveys.
CONTINUOUS MARKETING/UPSELL

By now it should be clear that moving from an on-premise to a SaaS model has some serious implications for marketing.

In essence you’re moving from transactional marketing (with the goal of getting someone to download a trial, or download the software) to continuous marketing (with the goal of showing them how to get more and more value out of your software). As your customers become more successful, you become more successful as well. If they’re not successful, and the first few early pioneers aren’t impressed, they’re not growing the account and they may end up canceling, which means no more revenue for your company.

The need for shared success often requires new people at your organization – including customer success managers and potentially customer marketing managers as well. Even if these people report into marketing or sales, their job is really post-sale, and they’re responsible for helping customers through deployment or implementation, making sure that users who are signed up for a service are properly trained and actually using it, keeping a pulse on customers’ level of happiness, and resolving any issues in a timely fashion. This requires a lot more frequent communication as well – no longer is it appropriate to send a welcome email and then nothing until it’s time to renew again. As SaaS model requires constant, direct interaction with your users.
Let's go back to the Netflix example. Say I got an email from them yesterday letting me know that the third season of Sherlock has finally been posted to Netflix. Now of course that notice didn’t go out to the whole world; it went out to those people that Netflix knows watched seasons 1 and 2. So they have a chance to touch me transactionally when something’s important. They also send me a monthly newsletter on important updates.

Interestingly, it turns out that **THE MOST IMPORTANT THING IN EVERY PERIODIC COMMUNICATION IS YOUR SUPPORT CONTACTS**, because customers are only likely to read the email when they have a problem and want to address it.

Therefore it’s important to repeat the phone number, email, or webpage where customers can get help, because they’ve definitely lost the welcome email by now. And so this transitions from a transactional model to a continuous model where different communications are sent based on customer behavior. If you’ve been through any of the Lean UX classes or Lean startup books, that theory applies very well here. With a large SaaS audience you can do lots of experiments like altering workflows, changing the messaging or calls to action, sending out new reports to influence behavior, etc.

Setting up a structured, trigger-based communication model can get complicated pretty quick, so start small and grow from there. I’d recommend segmenting your user base into, say, three categories; superstars, struggling, and inactive – and plan out your communications to address the needs of each group. For example, you could identify struggling customers based on support tickets logged, transactions that are dropped or canceled, abandoned shopping carts, etc. All of those are signals that you should reach out to help them along, as well as look at your systems for potential bottlenecks, broken workflows, etc. To identify the inactives, you should have access to login data to uncover accounts that haven’t used your software in, say, 1 month. Even though those customers may still be paying you money, you’ll want to reach out periodically to coax them back into the fold. Essentially retention becomes the essential revenue driver for the business as dollars come in each month. All of your communications, therefore, should be aimed at showing customers value and encouraging deeper adoption with your product.
NEW SERVICE METRICS

Often when software companies’ move from on-premise to SaaS, they find out they are missing an entire organization around operational efficiencies and the ability to run their software.

Remember back when you had IT organizations that you could ship your business to; and it was their job to get things running and secured? If it didn’t work you could put the blame on them because you weren’t the final deliverer. However, as you roll out your own operation staff you are going to have to find some of those same experts to handle things like uptime, response time, system capacity, support escalations, reporting, etc. Now you’re in the business of keeping customers running and satisfied – you’re now fully responsible for measuring and planning good service! If you haven’t lived through a P1 system down “Oh my gosh, SaaS is offline,” moment with your product, you either have a great operations team or it’s a very young product.
You’re going to need set up a parallel organization (generally to your development) because you’re going to have much clearer handoffs between the folks who build the code and the folks who run the operational systems to keep them up and running. Moving from on-premise to a SaaS model you should again expect to create an entirely new organization, new skills, and you’ll probably have to start with somebody pretty senior who knows what that really looks like because building that bottom-up is a challenge.

As a SaaS vendor, you’re no longer in a position to point the finger somewhere else. You’re running it, it’s yours, and customer satisfaction is your issue and regardless of the cause. If you’ve done something in your application that causes users to get confused then you need to fix it because it’s your work flow or your UX. Again downtime network failures, all of the challenges around Android where there are hundreds of different versions on different phones – if you’re putting out an app on the front end that talks to your software in the back end, you’ve got a lot of testing to do, and it’s entirely your problem. As we’ve seen with the major retailers, security holes are your problem so make sure you’re covering those as part of your responsibility in owning the customer experience from beginning to end.
CONCLUSION

In case you’ve missed it, software has already moved to the cloud, so if you’re building new software you need to make sure you have a real strategic reason to pursue an on-premise model or you’re probably making a mistake.

The consumer style view of the world has trickled back into the enterprise world. You can no longer get away with blue screens, ugly type, and confusing error messages because consumer expectations are much more demanding than they once were. Finally, to complete a successful transition, you’ll need to rearchitect your organizational structure to add the necessary operational skills in marketing and customer success.
ABOUT THE AUTHOR

Rich Mironov is a tech executive and serial entrepreneur and has been the product guy at 6 B2B startups. He approaches product executives on management teams and agile development organizations on how to really understand what customers want and what they will pay for. Rich founded the very first product camp which has now become a worldwide phenomenon and has been blogging about product management for more than a decade.

ABOUT LOGI ANALYTICS

Logi Analytics is the leader in embedded analytics solutions that enable software and SaaS providers to improve user experience and build product value. The company has a fully dedicated OEM team and over 10 years of experience helping hundreds of software companies to embed reporting, dashboards, and analytics. From visual design to feature selection and integration strategies, we’ve developed best practices for every situation and will work with you to make them part of your application. With more than 1,400 customers worldwide, Logi Analytics meets the analytics needs of organizations ranging from small businesses to Global 2000 enterprises. The company is headquartered in McLean, Virginia, with offices in Reading, UK and Stockholm, Sweden. The company is a privately held, venture-backed firm with investments from Updata Partners, GroTech Ventures, Summit Partners and LLR Partners. For more information, visit http://www.logianalytics.com/oem-solutions/embedded-reporting.