Converged CRS And PMS Are Finally *Here!*

Now What?

George Roukas
Contact: George.roukas@hudsoncrossing.com
www.HudsonCrossing.com

HUDSON CROSSING

Much has been written recently about the convergence of central reservation systems and property management systems. There are two reasons for all the writing: first, because it represents such an improvement in how hotels process their bookings and manage stays, and second because we've been waiting for over a quarter century for these converged systems to appear. Now that it looks like convergence is finally going to happen, let's consider its context and look ahead to convergence's wider implications:

1. Convergence is primarily enabled by **cloud computing technology**, which will enable lots of other things we've wanted to do for a long time
2. Convergence **won't stop with CRS and PMS**—it will allow companies with RMS, CRM, and other systems to converge those applications as well
3. Tech suppliers who develop these converged, cloud native systems intelligently, with well-structured and fully documented APIs connected to well-designed underlying microservices, **will create hospitality technology platforms** that can serve many kinds of applications – all using the platform's common services and data structures. (That's a mouthful, but we'll break it down later on.)
4. Some vendors will **evolve their platforms into full-fledged ecosystems** that support applications from multiple (and often competing) vendors; this will be a very good thing for almost everybody, but especially for hotels
5. As the ecosystems build, traditional applications **will decompose into separable modules that can be assembled** like Lego blocks to create functionality to address what each hotel, group and chain may need to differentiate their offerings to guests.

By the time we get to step 5, hotel technology will have more in common with smart phones and app stores than with today's hotel technology applications. Hotel technology will evolve from separate applications to platforms and eventually to ecosystems running discrete capabilities.

HUDSON CROSSING

Convergence is primarily enabled by cloud computing technology, which will enable lots of other things we've wanted to do for a long time

The same technologies enabling us to achieve converged CRS/PMS systems will continue to push us forward beyond convergence, and understanding that end game is critical to making the right choices now. This time around, the future is coming fast whether we're ready or not, and it's creating a new playing field where past leaders might be pushed aside to make room for new entrants.

Back in the 90's when we first heard promises of converged systems, we understood the problems they could solve (consistency, security, speed, etc.) and we waited with great anticipation for the day of their delivery. And waited. And waited. It's not as though the software companies didn't want to deliver, it just turned out to be much harder than they thought. Some of the early attempts didn't scale well, and the idea of scaling vertically by buying ever-larger machines seemed like too much work and way too much cost.

Then Amazon, Azure, Google, and Alibaba's clouds appeared, and cloud-native applications based on micro-service designs started popping up. Cloud computing allows you to scale up and down as much as you like, and only pay for what you use. Play by cloud rules and you could offload many of the technical challenges of enterprise applications[1] to someone else who was really good at it. By the way, the cloud companies were also very good at something that most other companies aren't: artificial intelligence. AWS, Google, and Microsoft all include AI primitives with their environments. And they're much better at AI than you are and likely ever will be. Does someone like Google's AI lead Jeff Dean or Stanford's Andrew Ng come to work at your company in the morning? Didn't think so--they're better at AI.

Convergence won't stop with CRS and PMS—it will allow companies with RMS, CRM, and other systems to converge those applications as well

The next question to ask might be: "If computing clouds let us converge a CRS and PMS, why stop there? Are there other hotel systems that would benefit from convergence?" And the answer is a resounding "you bet!" Hotels operate many interdependent systems including not just CRS and PMS but also revenue management systems (RMS,) and customer relationship management (CRM) systems, among others. Each of these systems has to interact with others to provide value to the hotel, and anyone who has worked with them can tell you that all that interfacing is much easier said than done. Each keeps its own data, its own core functions and services, uses its own methods for communications, and sees itself as the center of the hotel technology world. In the

[1] To be fair, there are lots of up-front issues and costs for companies moving to the cloud, which is part of the reason why so many developers are delaying. There are training issues, security issues, and architectural considerations that make it a non-trivial affair. It's a big change, but one that brings lots of benefits once you get it going. Think of it like running an auto race where you know you have to come into the pits to refuel: if you wait, you'll maintain an advantage while the others are in the pits, but delay too long and you could run out of gas on the track.

world of applications, that's life. But in the world of an open hotel technology platform, all of that changes. In fact, the notion of an application as a complete and unitary entity begins to melt away and we're really talking about capabilities that can be used independently or strung together in new ways.

Tech suppliers who develop these converged, cloud native systems intelligently, with well-structured and fully documented APIs connected to well-designed underlying microservices, will create hospitality technology platforms that can serve many kinds of applications – all using the platform's common services and data structures.

To see how the shift to platforms could happen, and especially why it will happen, have a look at figure 1 below. (Apologies to technologists who might cringe at the liberties taken below, but this will be useful for our discussions.)

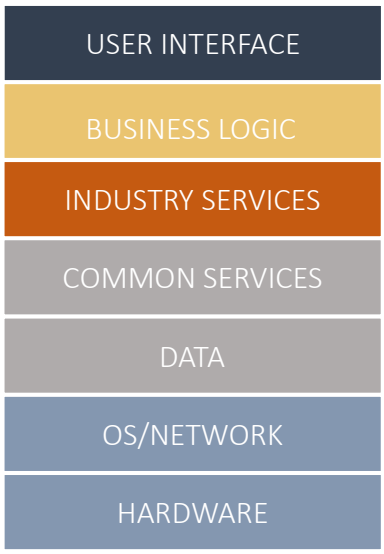

Figure 1: An application 'stack'

In this figure, the top layer, or user interface, is what you see when you use the application. If it were a PMS, you would see screens allowing you to check someone in or out, view a folio, assign housekeeping, etc. These are all parts of a UI for a PMS. The next layer represents all of the program code (business logic) that connects the UI to the underlying database and other services. Some of it is specific to your application, like the exact look and feel of your application, how it flows, whether it provides help, etc., and some logic is common to all applications of a particular kind. Keeping the previous example, every PMS has to allow a user to check guests in and out, and those are the kinds of functions we refer to as industry services in the stack, but the specific look and feel for how your application does it is part of *your proprietary* business logic. Below that are the service layers: this is where (1) industry specific services and (2) common infrastructure services live, providing common services and data structures that any application (in any industry) might use, like memory and storage. Finally, we have the networking connectivity and data layers. Today, common CRS and PMS applications have all of the elements from UI layer to hardware—a full stack.

When two of these systems have to coordinate with each other today, like a PMS and a CRS, they each keep all their own data and exchange pieces of it with each other. Those legendary 2-way, 1-way, and 1 1/2 -way interfaces allow applications to shuttle data from one stack to another -- while also shuffling the order of data elements around because no two applications store the data in their databases in the same way. It's errors in all this mess, and shuttling and shuffling data that

cause problems like inconsistent data, bookability errors, and security holes. It's a mess, but it's been better than nothing.

Enter the cloud-native system. If the software engineers can build out the upper layers using cloud-native methods, they wouldn't have to worry about the bottom four layers of the stack. The cloud would take care of it. And if you build your business logic through microservices, then you could share those services among however many applications you want—like a PMS or a CRS. There are several implications from building this way that open the door to lots of other good things:

1	2	3
If you're leveraging the bottom four layers on the stack in the cloud, ***you can automatically scale up or down however you need*** from your lowest period to your busiest period and the cloud would manage getting the resources for you. Got a big, new client? No worries: bring it on. Are there times when you just don't need much computing power? You don't pay for it when you don't need it. (This requires management but *can* be done.)	If you've got multiple applications (like that PMS/CRS combo we've been talking about) on your system, ***they would be using the same services and data structures.*** That means there is no duplication of common data between systems. *They're not talking to each other—from a data perspective, they ARE each other.* When the PMS makes a booking and creates a booking record, it uses the same industry-level services and the booking sits in the same database as a booking record made by the CRS, and it looks identical in structure. So no more consistency and bookability issues with the shared data.	Finally, since you only have to manage the top three layers, ***development can move much more quickly***. You've effectively outsourced the other layers of the stack to the cloud. You no longer have to deploy as many engineers on the 4 to 5 bottom layers and you can either save that money or, as most are doing, *redeploy it to multiply your engineers at the top of the stack where it creates the most valuable IP.*

HUDSON CROSSING

Some vendors will evolve their platforms into full-fledged ecosystems that support applications from multiple (and often competing) vendors; this will be a very good thing for almost everybody, but especially for hotels

If you follow the above, you can see that there is no need to stop convergence at two applications when you have more apps that can share the same services and data structures in a platform. Remember that a cloud provider would normally give developers services for the four bottom layers of the stack, and you definitely want to keep the top two for yourself because that's where your competitive IP resides, but how about that industry services layer? Provisioning that layer is what makes the hotel technology platform provider so valuable. By giving developers access to industry services like shopping, booking, check in/out, folio management, etc. the platform provider can set up platform-wide data structures and take a tremendous amount of work off the developers' plate. For example, developers would no longer have to design their own booking records or services to manage them because the platform would already provide them—and any system that sits on the platform could use the same booking records to do its work as well.

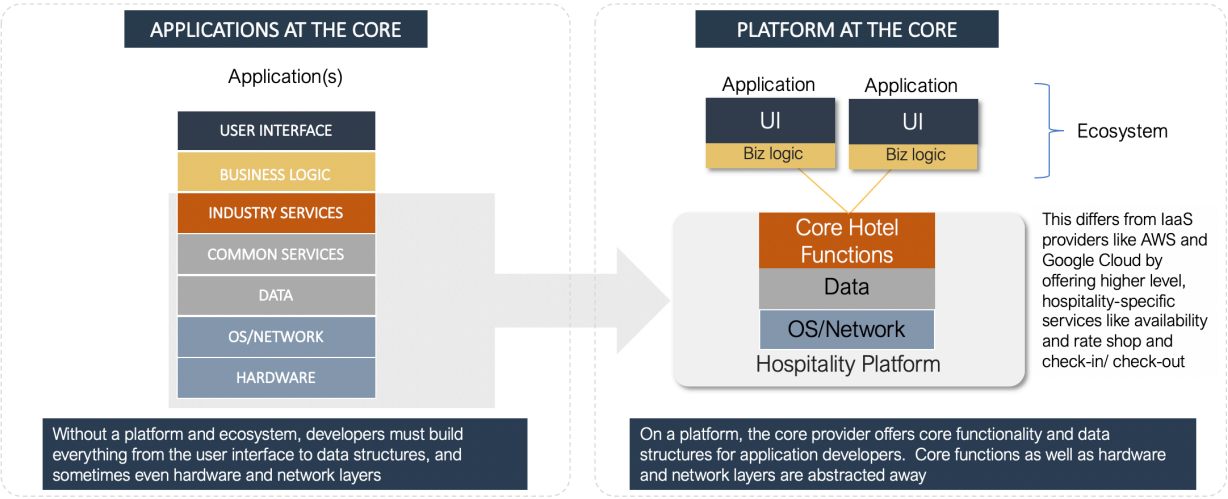

Figure 2: Traditional systems (left) vs. converged on a platform or (with multiple vendors) ecosystem

Imagine hooking up a revenue management system that shared booking data with your applications. Rather than having to export booking and pricing data to the RMS, processing it, and then sending instructions back to the CRS or PMS for pricing, that booking data is readily accessible to the RMS and the pricing decisions can be put right into the database where the other systems can use them. How about adding in a CRM system so you can get real time pricing based on what you know about each individual guest as they wander through the shopping path? Keep adding the relevant systems together and you multiply the value of the combination of systems, and that is what would make a converged hotel technology platform so powerful.

Where we currently have free-standing applications, each with a full stack of code, the hotel technology platforms (and we see some of them approaching readiness today) will provide a place where all of the applications can run on a common set of services and data structures hosted in a cloud—preferably a public cloud like AWS or Google. A vendor with multiple applications can either create the platform and migrate all of its applications into it, providing its customers with

the addition of a new services (for example, adding PMS for a customer who already has the CRS) with a small fraction of the migration headaches they would otherwise have to face today. imagine the implications of that. If you're a hotel that's using the Acme CRS on their platform and you want to switch to an Acme-connected PMS as well, there is virtually no data migration from your other PMS—that data is already in the cloud on the Acme platform. There will always be some configuration tasks when migrating to something new, but the Godzilla-sized data migration nightmare is reduced to the size of an iguana.

Where Acme's platform previously was gated by the speed with which they could build new applications, an open ecosystem has no such limits. When a stable and performant platform can run a group of competitive applications with multiple CRS, multiple PMS, multiple RMS, etc. (an ecosystem) you get a hotelier's dream: an 'operating system' layer for hotel technology, analogous to iOS for Apple iPhone or Android for other phones.

But, you might wonder, wouldn't it also be a software vendor's nightmare? What happens when all the best CRS (for example) are running on a single platform? Doesn't that mean they'd have to compete by lowering prices, creating a race to the bottom? The simple answer is no. To understand why, think back to the things that are being handed over to the platform: (1) all the networking and data components of a public cloud provider, and (2) the industry-level services and functions that are fairly commoditized by now, although not standardized. None of these provides any differentiation from one application to the next and all competitors want to provide them. In fact, the competitive value of the solution rests mainly in those top two layers of the stack: the user interface and the special sauce in the higher-level business logic. These are the things that companies can use to differentiate themselves from competitors and they are not part of the platform. We could argue that by joining a platform, a vendor gives itself some very important competitive advantages[2] in that it will be able to focus on the things that provide differentiation and value to its customers and it will be able to do that with more resources because it isn't wasting time on the lower layers of the stack.

One other aspect of the ecosystem that will be particularly compelling for software vendors is with what we'll refer to as mega-capabilities, the kinds of things that smaller vendors would find hard to do well or to do at all, and we can provide two examples here. The first might be something related to industry functions like shopping and booking capabilities. Expect that the big platforms will want to be best in class at this kind of functionality and will be sporting response times under 100 ms for shopping results, even under high volumes. That takes lots of time and money to get right and is likely beyond what smaller competitors could do.

[2] We could provide a reference that elaborates on how companies should try not to compete directly with rivals by trying to be the best at something; instead, they will be much better off by competing through differentiating their products and services to create new value propositions. But that would include virtually every book on competition produced since the 1980s. Instead, we'll quote Peter Thiel, co-founder of PayPal by saying "[Direct] competition is for losers."

The second type of mega-capability might be something like attribute-based shopping, or ABS. We've seen a lot of interest in development of ABS among vendors, and a parallel interest in hotel companies in the 18 months since we produced our white paper on it. We've also seen how difficult it has been for vendors to iterate with their designs to get it right—ABS is hard, and some parts of it will be very hard. The companies that have had the most progress are the ones with the deep pockets and determined management. ABS represents a type of capability that hotels will want to exploit and many vendors will find difficult to build, or they will build low-power versions as a marketing checklist item. Either way, we expect to see features like ABS as part of the draw to align with an ecosystem that already has it truly ready to use at scale.

To compound that, ABS performance will be driven by continuous deep learning algorithms that require lots of data to derive the attribute bundles that have the highest propensity of purchase for each guest. That takes time and lots, and lots, of data. That means that the hotels that get started with ABS sooner will have a distinctive first mover advantage, and one that is likely to expand over time. Not only will it pay for hotels to start testing and learning with ABS as soon as possible, but delays may be costly in terms of competitive disadvantage—and grow more costly over time as the early adopters extend their leads.

As the ecosystems build, traditional applications will decompose into separable modules that can be assembled like Lego blocks to create functionality to address what each hotel, group and chain may need to differentiate their offerings to guests.

Another interesting effect of ecosystems will be that as they develop, the structure of applications will 'melt' into components structured to deliver capabilities that can be assembled like Legos to create the functionality hotels need. In his excellent book "Smart Business: What Alibaba's Success Reveals about the Future of Strategy," Ming Zeng, the former chief of staff and strategy adviser to Alibaba Group, categorizes the various components of an ecosystem using geometric terms. He describes the platform as a *plane*, applications (or collections of related capabilities) as *lines*, and individual capabilities and services as *points*, and each has a role to play. The value proposition of a plane is to connect related parties, the value proposition of a point is to sell a function or capability, and the value proposition of a line is to use the capabilities of points and planes to provide products and services.[3] The main thing about this categorization is that all three are required to make an ecosystem function properly. In the hotel technology space, we can think of the collections of capabilities in traditional hotel systems as lines. Many will split off capabilities they do well (like an IBE or housekeeping module) to create individual points or other lines and many developers will create completely new points (like intelligent agents that monitor shopping in real time to coordinate the actions of shopping, revenue management and CRM to provide real-time, personalized recommendations) to provide capabilities we don't have today.

There are also some interesting second order effects from a competitive standpoint. Remember that you can have multiple applications of the same kind running in the ecosystem and using the same data structures underneath, so a hotel would be able to run multiple applications similar

[3] Zeng, Ming. Smart Business. Harvard Business Review Press. Chapter 6.

functions in order to compare them. For example, suppose your CRS vendor provides an internet booking engine (IBE) and claims that they have a higher conversion rate than anyone else. Now suppose you are approached by another vendor who only makes IBEs and claims theirs is superior. Today, a comparison would have to be run against the same time last year or concurrently, but with different hotels. There is no opportunity for a true A/B test. In an open platform ecosystem, you could operate your CRS vendor's IBE and carve off some traffic to test with the other vendor's IBE and compare the results in a true A/B test. You'd have a lot easier time determining which converts better, and you might find out that one converts better in Europe while the other does better in Asia—and keep both!

Another example might be if you've ever suspected that one of your vendors (for example, an RMS vendor) is selling more sizzle than steak, it would be possible to set up an A/B test and find out whether the claims in that slick new marketing brochure are real or not. Just plug in the second vendor and start comparing the actual results. If you're a vendor who has relied on marketing more than product, your days are numbered.

In a similar vein, competitively speaking, the ecosystem makes it very difficult for vendors who have kept customers by setting up 'moats' to retain those customers. For example, an application might keep competition at bay by being the only game in town that can do all of the things a hotel chain needs. Maybe they are the only one with the breadth of features, or the service network that stretches across the globe. They win today because they're competing against each of the other applications individually. In an ecosystem, however, they would have to compete against *all of the other ecosystem applications as a group*. Hotels using an ecosystem could select several applications with each providing a different piece of what they need, confident that all of those applications will use the same services and data, and thereby avoid being held hostage by any one vendor. Liberating, eh?

What does a platform look like and how do I find one?

Finally, how can you spot a real platform in the wild? Just about everyone claims to be a platform of some type or other—it's a great buzzword that's lost most of its legitimacy through overuse. But we believe that here, as with other successful ecosystems, *there will likely be only 3 or 4 heavyweight vendors that can provide a "plane" that attracts a critical mass of solutions (points and lines) and hotel companies*. It's important you choose someone likely to be on that short list. Here are a few things we're looking for when recommending options for our clients:

- Large developers, or smaller entities owned by larger companies.
  Building the kinds of platforms we're talking about costs tens of millions of dollars. By our estimates, candidate companies we follow have spent in the vicinity of $50-100 million in development costs and they're not yet finished. There are some smaller guys who have done some impressive development, but they're unlikely to be able to provide the breadth and quality of features at the scale needed to be a surviving platform or ecosystem player for complex independent hotels and larger chains.

- Public, cloud-native applications with rich feature sets.
  Platform applications need to be feature rich, industrial strength and cost effective. That means they have to be able to scale quickly without having to build business cases with the finance team and purchase orders for new equipment (remembering to file the goldenrod copy in the third cabinet on the right) so they finally come online 3-4 months later than needed. That's a very pre-AWS process that won't cut it in the new world. Hospitality technology companies will need to outsource those worries and get what they need upon request from the cloud vendors, and they'll need to do it at a competitive cost. If you're thinking you can operate a cloud better than Amazon, Google, Microsoft, and Alibaba you're in the wrong business!

- Forward looking, with a fundamental understanding of what's next in hotel technology
  Some of the vendors who are sprinting to create a next gen platform are ignoring advances like ABS to get out the door faster. That strategy will enable their customers to get into a platform of sorts that provides some of the advantages we've outlined above, but they'll be closed out (or far away) from all of the biggest benefits that ABS, and other mega capabilities, bring. Retrofitting a data model to be ABS-aware is a long and expensive road that can put you far behind the vendors who did the right design work in advance. Again, many of the greatest benefits from implementing ABS (and wiring it into your revenue management, CRM, and distribution systems) will accrue to those who drive ABS options with deep learning systems that get better with more data over longer time periods. It's a flywheel that improves the whole system over time. Taking a knee for a few months or years while you try to look at it from every angle could put you behind others, so jump in with a mindset of experimentation and learn as fast as you can. This is no time for guiding your company into analysis paralysis!

In Walmart's early days, Sam Walton faced a daunting competitor in Kmart: the undisputed leader in big-box retail in the US. Walton understood that competing directly with Kmart would be disastrous, so he made two key competitive moves. The first was to build his stores in secondary cities so he wouldn't have to share customers with Kmart in the major cities it favored. This allowed Walmart to survive. But the second move was further reaching: Walmart's innovation was to reject Kmart's policy of running each store independently in favor of connecting all of the Walmarts to a central hub and operating them as a network. The strategic impact on operations was significant. For example, when innovations like bar code scanners arrived in the retail industry, Kmart saw them as a way to avoid having to put individual pricing stickers on products, which reduced individual store errors and labor cost. Walmart did the same, but it also fed the data into a centralized order management and logistics system that analyzed which products were selling where. That allowed the company to negotiate prices, purchase goods, and ship inventory far more efficiently and at lower cost. In effect, Walmart had changed the unit of competition from the store to the network, and it weaponized that network against a much larger competitor. We all know how that story played out for Walmart—and Kmart.

We believe platforms will change the unit of competition in travel technology from the application to the open ecosystem. Developers who decide early where they will play in the ecosystem and weaponize their assets in the new environment will have the same opportunities as Walmart to defeat the travel technology Goliaths of the last few decades. Developers, hoteliers, and guests will all benefit handsomely.

HUDSON CROSSING