RIPARIAN

Are you using the term "Agile" correctly?

There are a host of technical terms that get thrown around when discussing software implementations and outsourcing solutions. Understandably, these concepts are often misused or confused, which can improperly influence our decision-making and ultimately our allocation of resources. In this article, we will explore some key terms and how we can properly apply them to business planning and vendor management as we move into the modern digital age.

Modern Application Architecture ("MAA"). Prior to recent advancements in technology (e.g., cloud platforms such as Amazon Web Services) and a growing demand for nimble contracting strategies, single-platform or single-solution providers were the go-to choice for the pharmaceutical industry when tackling commercial and government contracting. As the world evolves, however, so must our thinking around the solutions that support our business functions. Recognizing the value of recent advancements, when Riparian embarked on the journey of developing our software solutions, we centered on some key concepts that were critical to our success. One of them is MAA which is a foundational concept used for flexible and Agile software development, giving organizations the ability to reduce risk and pivot. MAA stresses certain process characteristics such as organizing around business capabilities, componentization, the ability to interface with other platforms, and Agile development. Having explored these concepts from a foundational business perspective in developing our software solutions, we also believe the concepts should be applied more broadly--- beyond software to the process of outsourcing commercialization services for the life sciences industry.

Componentization. Again, historically, organizations looked to single software-platforms or single-solution providers to implement or outsource their revenue management/ contract management requirements. From a contracting perspective, this makes a lot of sense as there are fewer organizations to manage and contracts to negotiate providing a "one-throat-to-choke" scenario. While streamlining contracting solves one risk, it also creates a plethora of additional risks. First, and most obvious, this "all eggs in one basket" approach creates a single point of failure, and therefore, failures during the implementation (or "modifications" to timing) also put the entire solution at risk. Additionally, when platforms or services lack interoperability (e.g., they only work with that single organization), a manufacturer's ability to

pivot falls considerably—the technical term for this is called "vendor lock-in," which causes the manufacturer to feel "trapped." Now the manufacturer is reliant on that single organization with a product that may not meet their requirements, that took way longer to implement, for a price that is no longer worth the investment. As a consumer, we would never tolerate this setup.

We like our Apple products, but we want them to work with our Android applications just as well. We want to buy the best products for the money and not be fully committed to one company. "Componentization" is a strategy that allows us to find the very best version of each component in the process or service line, rather than using one source with varying quality along the process, thus weaving together the optimal solution for the organization. If one of the components no longer works or is not meeting your organization's standards, you can pivot without affecting the entire ecosystem. We see this in the context of breaking out the software solution into pieces, as well as compartmentalizing vendors for outsourced services.

Plug and play. Most single-platform or single-solution providers do not work well with others. These providers would have you believe that they are "best-in-class" at everything and that their solution can solve all problems. But it's rarely the case that a single-platform is best in class for all functions. Unfortunately, single-platform/solution dominance is not based on subject expertise or the strength of features, but in the leverage the platform has over an organization once it has invested in the implementation. If you are not satisfied with a portion of the software, or a particular application, module or service, now you have limited options because the all-or-nothing solution becomes too expensive to move off all at once. This leverage may be expressed through poor customer service, less frequent feature updates, and ultimately strained relationships. This single-platform dominance way of thinking was extremely prevalent during the hardware and software wars of the 1980s with IBM, Microsoft, and Apple fighting for supremacy or a complete monopoly. Of course, things have changed since that time and now you can have Windows work on your Mac or iTunes work on your Windows machine, and we can all agree this is a benefit to the consumer.

Interchangeability and interoperability have become a cornerstone of modern consumer-based applications, and you see great applications utilizing application programming interface ("API") plugins to allow the use of various programs seamlessly. These programs lack the leverage over the customer of a single-platform system, and it forces the developers to create better user-based applications for consumers. At Riparian, we believe that enterprise applications and outsourced service solutions should work the same way. With a client-centric approach to solutions, clients should stay with each application or outsourced service because it is the best application or service and not because it is stuck with the challenges of migrating off an expensive single-platform, when the individual solutions are not optimal for its needs.

Agile. "Agile" has become a business buzzword often overused in communications but underutilized in practice. While there are volumes written on the concept of Agile as a software development project management process, which we will not recap here, some of the key characteristics include maintaining user focus, coordinating cross-functionally, welcoming change, and embracing simplicity. Additionally, the Agile concept adopts a foundational theory of evolutionary development, continual improvement, failing often and failing small, in order to ultimately achieve success. This contrasts significantly to the "Waterfall" principal of development which originated in manufacturing and construction, whereby everything follows a logical sequence of activities taken through the software

development life cycle ("SDLC") cascading down (like a waterfall) until the final deliverable is presented.

While there are advantages to the waterfall process (e.g., tracking to defined steps), one significant disadvantage is its inflexibility; it is very challenging for a vendor to pivot when requirements change or are missed in the beginning. While the Agile process allows for failures that are identified early but remain small, the waterfall process will not reveal failures as frequently or clearly, and by the time they are discovered, they are significant. The waterfall process inhibits the ability to pivot. In the consumer world, large software companies such as Google embrace Agile, and are therefore able to pivot, this concept has not caught on with many software vendors focused on the life sciences industry. Some life sciences vendors tout the term Agile for software implementations, but mainly just use the term to allow for delays. The key to understanding the difference is focusing not on the implementation, but whether the underlying solution itself was built on an Agile platform that allows for pivoting when circumstances and/or requirements change.

For many, historically chosen solutions are not user-focused, do not satisfy cross-functional needs, are resistant to change, or are overly complicated and difficult to use. This means they ultimately fail clients in big ways costing them millions of dollars in overage fees and months, if not years, in delays. When considering vendors for software or services, question how the vendors are going to meet your needs. Understand that there are going to be unforeseen changes in the underlying premises of the solution and/or failures; consider where those failures or changes are likely to occur and look at how the vendors are going to ensure that changes or failures will be small and identified quickly so that they will not impact the overall time or budget. Question how those vendors are nimble enough to welcome change and how they embrace user-focus, cross-functional coordination, and simplicity. In the end, is your vendor Agile?

In summary, understanding the terminology and evolution of recent software development concepts will help reduce risk and increase flexibility ultimately saving your organization time and money when implementing software or outsourced solutions. Incorporating the key concepts of MAA, such as componentization, plug and play and Agile, is not a silver bullet by any means. It might mean hiring more than one vendor to obtain the best-in-class solution and it will likely force more cross-functional and cross-team coordination. But team management is a skillset your organization most likely already owns. So, if you want to prioritize reducing financial and organizational risk and gaining the flexibility you need to pivot in these fast-changing environments, then incorporating MAA concepts is likely the right direction for you.

References:

1. Devops Digest. Modern Application Architecture is Critical to Business Success. October 2018. https://www.devopsdigest.com/modern-application-architecture-is-critical-to-business-success
2. Mary Schacklett. 5 Ways to Avoid Vendor Lock-In. October 2018. https://www.techrepublic.com/article/5-ways-to-avoid-vendor-lock-in/
3. Andrew Powell-Morse. Waterfall Model: What Is It and When Should You Use It? December 2016. https://airbrake.io/blog/sdlc/waterfall-model

About the Author

David S. Chan

Co-Founder & CEO, Riparian LLC

David S. Chan is the Co-Founder and CEO of Riparian LLC, which provides software, outsourcing and consulting solutions to Life Sciences manufacturers. David has overall responsibility for the strategic direction, product development, sales, operations and specific client initiatives. David brings 20 years of experience leading revenue management and contracting teams in the life sciences.

About Our Contributors

Susan Dunne

Senior Director, Consulting and Managed Services

Susan Dunne is the Senior Director and a member of the Riparian leadership team based out of the Virginia / Washington D.C. area. Susan is a licensed attorney with over 25+ years experience working with life science manufacturers providing support for their interactions with federal pricing programs. Susan focuses on GP compliance issues and strategic decision-making and has long standing relationships with industry leaders in government, outside law firms, and industry executives in large, medium and small pharma, both branded and generic.

Contact Us

Email: info@riparian.com
Website: Riparian.com
Phone: 646.809.4201