Using Functional Project Phases to Manage Complex Software Projects

Author: John Moody

Claps: 104

Date: Oct 30

Email is complex. To be more specific, *delivering* email is complex.

When you address your email and hit â€œSend,â€ the dance begins. Talk to the recipientâ€™s mailbox provider. Make sure your DKIM keys are in order so the mailbox provider knows you are who you say you are. Send the email â€” does that inbox exist? Is it full? Does the email content look like spam? Is it coming from an IP with a reputation for sending spam? Is the sender sending too fast? Tell them to back off and try again later.

Multiply that times the billions of emails Klaviyo sends every day, and you begin to see the complexity that MTAs (*message transfer agents* â€” the systems that deliver email) have to deal with.

So when we at Klaviyo decided to build our own MTA, it was a project unlike any other. We had to make sure all the complexity of email delivery was accounted for in our system, which in a burst of non-creativity, we christened â€œKMTAâ€ (for â€œKlaviyo MTAâ€). It was ultimately a challenge of project management. I had just joined Klaviyo as the engineering manager for the KMTA project. Having led complex software projects in the past, I knew that we needed an approach that allowed us to incrementally add functionality yet was fully testable at every phase of development.

The KMTA team accomplished this by defining a series of project phases during which the nascent KMTA would have a working subset of the total functionality required. We began with Stone, which was basic sending functionality. When we could send an email through the system and get it into an inbox, we had completed the phase. Bronze followed, and engagement tracking and deliverability event handling were added. Iron came next, and scalability was added in. Steel brought the handling of feedback loops and the monitoring tools weâ€™d need in production. Weâ€™re currently in the Titanium phase, where weâ€™re running live traffic through the KMTA system to establish IP reputation.

Defining project phases enabled us to focus on specific pieces of the overall system while ensuring that the end result of each phase met a specific quality and performance bar, thus ensuring that bugs and performance problems were caught early on. It also gave engineering leadership a way to see our progress in something more tangible than some opaque metric like â€œnumber of tickets completed.â€

It also allowed us to experiment. When building our engagement tracking system, we decided to write it in Go, even though Go was not in use at Klaviyo. If it didnâ€™t work, we could always rewrite it. We treated it as a pilot project, building out the changes weâ€™d need for our CI/CD pipeline, setting conventions for logging and test libraries, and documenting as we went. It turned out to be a huge success, and now other teams at Klaviyo are embracing Go.

Of course, there were setbacks. During load testing, we discovered that our Pulsar-based messaging system (the system communication type, not the email type) wasnâ€™t configured to

handle the large messages we were pushing through it. The phase was halted, some collaboration with our Platform Services team ensued, we made some adjustments, the system was tuned, and work continued.

If you want to try this approach, I recommend the following steps:

1. Figure out the components of the system to be built. (By â€œcomponentsâ€ here I donâ€™t mean physical components, although they could be that. Rather, I use â€œcomponentâ€ in the sense of â€œlogical groupings of functionality.â€ For KMTA, our components were things like â€œengagement tracking,â€ â€œsystem configuration,â€ and â€œscalability.â€
2. In a spreadsheet, create a matrix with your components from step 1 as row headers, and then label the columns with your project phases (use whatever naming scheme makes sense for your organization). We referred to our rows â€” representing logical groupings of components â€” as â€œswimlanes.â€
3. Assign functionality to each phase in the intersecting cells. If the swimlane is fully completed by the work, color the cell with a green background. Some swimlanes will likely be blank for a particular phase, and thatâ€™s OK. You wonâ€™t work on every swimlane in every phase â€” indeed if you have many swimlanes (we had 16), I recommend against it. The point is to move the project along during each phase, making sure that each swimlane ends up green by the final phase.
4. Check for dependencies between your tasks and swimlanes, and make sure that pieces are ready by the phase theyâ€™re needed. Shuffle as necessary.
5. Important: The next phase doesnâ€™t begin until all benchmarks for the prior phase are met.

	A	B	C	D	E	F	G
1	**Phase**	**Sandbox**	**Stone**	**Bronze**	**Iron**	**Steel**	**Titanium**
2	Theme	Dev environment testing / exploration	First production deploy; integrate with test Klaviyo accounts	Monitoring, first pass at engagement tracking, deliverability, & config mgmt	Infrastructure to route traffic to KMTA; expanded engagement tracking; validation of core system capacity through load testing	Complete infrastructure components; Begin serving production traffic to warm production IP's; Finish buildout of monitoring / alerting tooling	Full operational readiness for BFCM 2023
4							
5	Email Authentication					<10 DKIM keys (manual rotation)	
6	Ingress API		Use account hot setting to divert traffic to KMTA		New multi-modal traffic splitter replacing current traffic split logic		
7	Resilience / Availability		Initial infrastructure (MTA / HAProxy) stood up		RFC for persistent EBS volumes	EBS persistent volumes; Elastic Network Interfaces -> HAProxy integration	
8	Scalability				Build load testing harnesses for all KMTA components		All components load tested and verified to scale
9	Managed Configuration			Build out configuration tools to manage GreenArrow / HAProxy configuration			All configuration tools fully operational
10	Campaigns Integration		Limited (via account hot setting)		Full (using multi-modal traffic splitter)		

Simplified and redacted version of our KMTA matrix

KMTA is transitioning to operational mode now, so the need for a phased approach is behind usâ€¦for this project, at least. And itâ€™s a good thing, because weâ€™re running out of metals!

(Adamantium? Vibranium?) But for future complex projects, itâ€™s a project management technique Iâ€™ll rely on â€” itâ€™s proven itself admirably on the KMTA project.