DRIVER	APPROVER	CONSULTED	INFORMED	APPROVAL DATE
Dobromir M…	Robin Singh Saurabh Lad…	Reid Senescu Saurabh Ladha Justin Winckel eng-staff@do… product-all@…	eng-all@doxel.ai	2022.01.07

Executive Summary

EPDR is moving to a 6-week execution cycle, with a 2-week planning / engineering-investment step between execution steps. At the beginning of every 6-week “build cycle” we will publish what projects we are executing, the goals for each project, and if Engineering has committed to the goals, consider them “stretch”, or have no idea and are approaching them as R&D. At the end of every Cycle we will publish how we did on commitments, goals, and R&D.

This replaces our 13-week (quarterly) planning & execution framework we’ve been using thus-far.

We will reserve some amount of time (10-20%, depending on the team) in each build Cycle for small features, bug fixes, etc. In general, we will say NO to anything taking longer than a day or two once the Cycle has started and ask that it gets prioritized following this process for the NEXT Cycle. This means, worst-case, it could take 8 weeks for work to reach the “execution” phase. This is faster than today’s 13 weeks which should make us more agile.

While not all “engineering investment” can be firewalled into the 2-week cycles we will attempt to minimize the distractions from building Product during the 6-week Build Cycle, and minimize distractions from improving our productivity during 2-week Engineering Cycles. The way we will do this is by having the Head of Product be the main decision maker for what gets prioritized into our 6-week Build Cycle, and the Head of Engineering be the main decision maker on what gets prioritized into the 2-week Eng Cycles. Since we expect everyone to be aligned to business goals this is mostly a formality to ensure quick decision-making.

We will run the first Cycle starting on February 24th, 2022, with the first prioritization pass happening Feb 10th, 2022. This should give us enough time to prepare for the new planning format. We will be doing an abbreviated planning session and ad-hoc sprints for work before then.

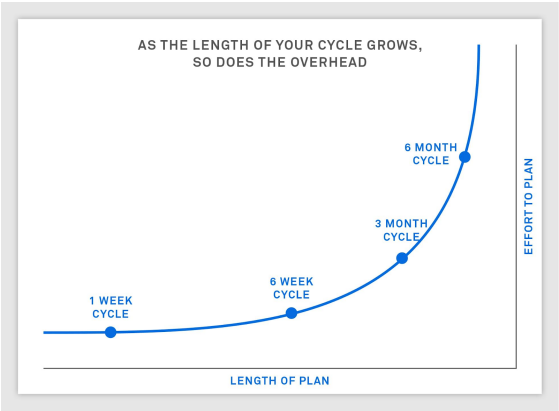
We need *executive support* to ensure **that Cycles do not get disrupted** by fires too frequently. Disruptions will prevent us from getting the focused time we need to ship predictably.

Motivation

Robin & Saurabh have asked us to build a predictable product delivery engine. As a startup selling to big Enterprise customers in very long sales-cycles it makes strategic sense for us to forecast 3-6 months ahead what will be built so Sales can start pitching it early and not get the company into promises we can't keep. As a startup we still need to maintain a level of agility to be responsive to things we learn, customer needs, and new opportunities. We believe our current 3-month OKR planning approach isn't meeting these constraints.

Per feedback from several discussions with Engineering and Product leads our 2021Q4 planning was unnecessarily hectic. Our top-down direction came quite late, after bottoms-up planning had already started, which led to a last-minute re-drawing of teams and efforts. This wasted both calendar time and people's time, probably on the order of 2 weeks for the former and 1 week for the latter. We also invested around ~25 hours of Staff's time (Saurabh/Robin/Dobromir/Reid, with some of Justin and Khoi's time thrown in), which is a very significant effort.

In my experience planning does not scale linearly with the length of the execution time which contributed to this feeling of "lots of work but so-so plan."



Other than time spent planning we've seen lots of pain points we can fix with a better planning process:

1. We are not very nimble in reallocating engineers today; the quarterly planning process means teams have committed ~3 months ahead and changing the plan requires re-planning. The extra friction (and lack of process) means it doesn't happen organically and we've had low-ROI work happening for ~months (e.g. Budget team in 2nd half of Q4 2021).
2. There are no enforced prerequisites before Engineering begins building which has led to low accuracy commitments, e.g. our Facilities Viz launch dates. Engineering feels a lot of pressure to make commitments while the deliverables are still being fleshed-out. The impact to the business is missed deadlines and disappointment across functions.

3. It's not clear whether a particular project is bottlenecked on Design, Product, or Engineering before execution gets started. Without clarity across all projects, the executive team may not know how to help the company remove bottlenecks.
4. Projects that require engineers working *across* vertical teams have no process for getting staffed. Most engineering investment projects (D2.0, B2.0) need to be "rolled into" a product launch in order to actually get built, which makes the product launches feel slower and doesn't give visibility to the engineers making the foundational investments.
5. Engineering doesn't have visibility into what Product & Design are thinking ~months ahead which may lead to engineering effort waste. A lot of our quarterly planning goals define the *goals* without defining the what & how, meaning that execution is supposed to start without clear work that needs to get done (e.g. Schedule Q4 goals).
6. Engineers don't have clear visibility into how the work they are doing affects the company's goals and success. We've had engineers leave due to a feeling of low-impact work (Deb in Q4) and we've heard feedback from others that they don't feel motivated about their impact.
7. It's not clear to individuals how to get their ideas prioritized and executed, which leads to a lot of top-down planning without leveraging our bottom-up collective intelligence. This also lowers individuals' feelings of ownership since work feels "assigned" rather than "volunteered for."
8. We've spent a lot of time fleshing out work that we do not end up executing, wasting our efforts in planning. E.g. this happened with our User Management exploration in Q4. Priorities are often not clear leading people to invest efforts in lower-priority work.
9. Lack of transparency on why certain projects are prioritized and others are not. The decisions being made are ad-hoc and often not recorded anywhere. It's unclear, for example, to Ops on why we may not be investing in more tooling - the tradeoffs (what we are investing in instead) isn't clear.

Proposal

We propose to mostly eliminate Quarterly Planning and switch to a 6-week project cycle (inspired by Shape Up from BaseCamp). All projects would need to be scoped to fit into a 6-week timeframe by Design/Product/Engineering and have **explicit** commitment from Engineering that we can ship something useful in those 6 weeks. Larger projects would be broken up into 6-week milestones; smaller projects/features/bugfixes could be grouped together and prioritized as a "bag" of deliverables. Proposals would go through a formal process to ensure they are "shovel-ready" and get final go-ahead from Reid/Dobromir (or R&S, if they want to run the process).

Teams do not have to wait 6 weeks to release features, nor do they need to release something externally-facing at the end. They do need to be ready to "put a wrap" on the work at the end of 6-weeks since they should expect that another prioritization pass will occur and any follow-up on the projects may be cancelled or delayed. That means having all the code checked-in,

reasonably tested/documented, possibly released to staging, etc. We want to avoid wasted effort so cancelling or pausing work on a project will hopefully be the exception.

Process

Phase Gates

We will start executing all projects using the Phase Gate approach proposed here: Phase-Gates. Phase Gates will align with our Cycles below.

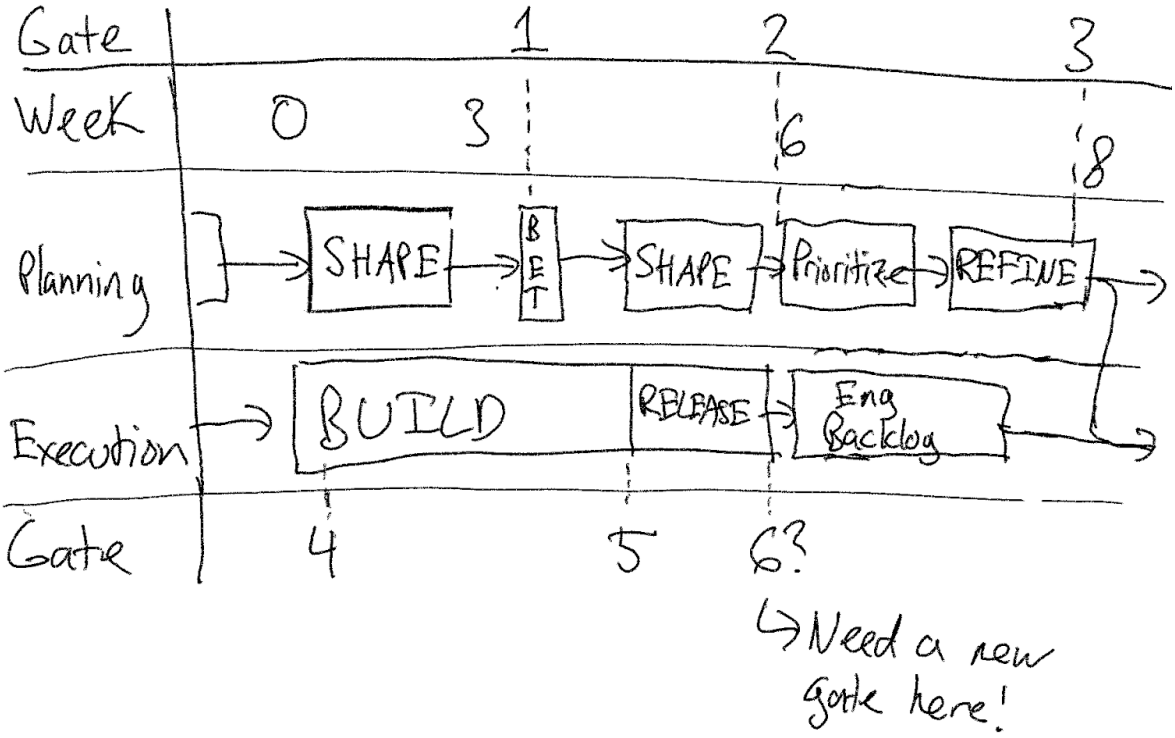
Cycles

Execution teams (Engineering + Product + Design) will have 6 weeks “build” efforts called a Cycle. This will consist of 3 2-week Sprints like we run today, with the 3rd Sprint really focused on wrapping everything up. Engineering teams **still need to support existing products** during a Cycle and so they should plan for up to ⅓ of the Cycle time on support and ⅔ on feature development. “Support” needs to be enforced by the Engineering lead - P0 bugs (customer-affecting) and “small” improvements (~1 eng-day) can make the cut. Bigger features should go into Cycle-planning and not be “snuck into” an existing Cycle.

During a Cycle, Product + Design + engineering leads + Executives will be Shaping the work for the next cycle. This means fleshing out the customer research, designs, PRDs, and TDDs to the point that engineering feels comfortable committing to “yes, we can build this in 6 weeks, with these people.” Shaping maps to Phase Gates 0-2.

At the end of a Cycle we would go into full planning/prioritization mode (“Betting”). Engineering gets 1 Sprint for engineering investments: the Engineering Managers are the decision makers on what gets done during these sprints and will have their full horizontal team available. Note this means that any bugfixes, feature requests, etc would be delayed by 2 weeks (unless an absolute P0) and need to make it into another Betting Table.

The timeline looks roughly like this



Why 6 weeks / 2 weeks?

There are several constraints that led us to pick 6/2 week “build” cycles:

1. Large features require multiple Sprints to build and about a Sprint to QA/release. If we tried to lower our Cycle time to 5 weeks or 4 weeks we’d be unable to fit “large” requests into the pipeline and would have to break a lot of work into “medium” sized pieces which would span 2+ Cycles. That would decrease the usefulness of Cycles as the main prioritization system: we’d basically be signing ourselves up for 2+ months of work up-front, which is where we are today.
2. Estimates are always wrong. If we want predictable delivery, we need some buffer within a Cycle to catch up if the unexpected happens. The business is likely to ask for the same set of features and try to squeeze it into the shorter timeframe, removing the buffer, increasing delivery risk, and leading to more missed expectations.
3. Engineering investments in productivity, more reliable systems, documentation, testing, etc are always needed. Usually, teams struggle to get this work prioritized against urgent business needs which leads to product delivery actually slowing down and unhappy engineering teams. In my experience, a healthy engineering team will spend about 20-30% of their time on investments. Unhealthy systems sometimes go up to 50-100%

of engineering. The work always needs to be done in a long-enough timeframe; by building it into a regular cadence we'll have both higher product delivery predictability, and better morale within engineering.

a. If we don't have high-priority work in engineering we can always choose to prioritize smaller product deliverables (bug fixes, features, etc) while doing Cycle planning.
b. Once we've migrated to D2.0 and have a fairly stable product roadmap + engineering infrastructure, we could lower this to 1 week or alternate between 2 weeks/1 week.

Shaping

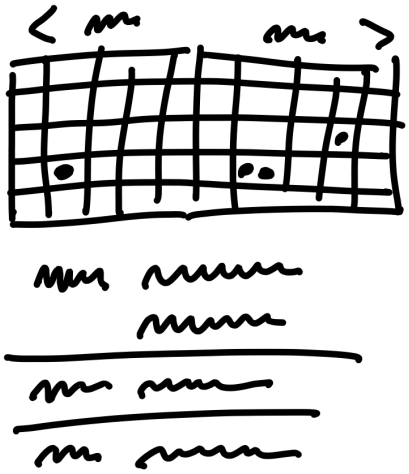
The goal of "Shaping" is to get a "good enough" project proposal. It doesn't need to be full designs or a PRD, unless the work is so complex that this would be necessary for engineering to commit. During the 6 weeks of Shaping, Product + Design (and optionally Engineering) should be focused on customer research, testing out designs, conversations with engineering on what is possible, etc. The "how" of Shaping will be very different team-to-team, but the output needs to be the same:

The *minimum* output of Shaping to get to the "Betting Table":

1. A **Pitch doc/deck:** this is an easy-to-consume (1-2 pages) thesis about a problem we should solve, the business case why we should solve it, the team needed to build it (individual names), and a **sketch** of what can be fit into a **single Cycle**. This needs to be approved by one PM + one Eng Lead to go to the **Betting Table.** This is the output of Phase Gate 2.

   Template that **must be used** is here.
2. A **Sketch**: this is a "fat marker" (not even a wireframe) view of how the product should look. For non-UI efforts this can be a simplified system diagram, data flow diagram, etc.

Example for a Calendar app:



Most pitch docs/decks should come *from the teams themselves*, but anyone at the company can put them together and add them to the Betting Table.

Note that the pitch deck needs **both** a PM and Eng lead to sign-off before it is considered ready. This is to help minimize surprises in the project during execution. If the work is particularly novel or difficult (e.g. Facilities Visualization) there may need to be a lot more details for a commitment to be produced - or a lot more scope reduced to limit the risk. The PM leading the effort + the Eng lead responsible for delivering should negotiate all of this *before* reaching the Betting Table and be aligned on what can be delivered in 6 weeks.

Prioritizing

All the in-progress pitches will be recorded here and marked as Ready in order to get onto the “Betting Table”. The prioritization process will go as follows:

1. Day 1: large meeting with all PMs/Eng leads. Go through all the pitches and mark them as:
    a. Yes: definitely execute it this Cycle.
    b. Maybe: likely do it this Cycle; decision will happen next week at the final meeting.
    c. RFI: there is some information missing preventing us from making a decision. The team has a week to get back with the additional information, or if it feels like there is a lot of work left, decide to bump this to the next Betting Table.
    d. No bet: don’t do it this Cycle, but leave it on the Betting Table for next Cycle.
    e. Cut: don’t do it this Cycle and don’t revisit it again; unlikely we ever want to do this.
2. Day 2-5: teams start executing on the priorities.
    a. Bet: finalize designs, PRDs, TDDs, etc. Start on Phase Gate 3.
    b. Maybe: ditto.

    c. RFI: start working on the questions, or wait until the Cycle starts if bumped to the next Cycle.
3. Day 6: Final prioritization meeting.
    a. Review all the designs/PRDs to make sure the team is aligned on what we are delivering.
    b. Make the final decision on what is going into the Cycle. Categorize all the Cycle goals as:
        i. Committed: Engineering is committing to shipping this by the end of the Cycle.
        ii. Stretch: Engineering believes it can ship this by the end of the Cycle, but there are remaining questions (well-documented) that prevent a strong commitment.
        iii. R&D: Engineering doesn't believe it can ship it by the end of this Cycle, but we still believe we should kick off the work.
4. Day 7-10:
    a. Finish Phase Gate 3 and be ready for implementation.
    b. Engineering leads start preparing new execution teams, if needed.
5. Day 11: Kick off the new Cycle.
    a. Finish Phase Gate 4

Execution & Reporting

We will continue to execute using the Sprint Process defined at [APPROVED] Product Engineering Sprint Process. For now, we'll report bi-weekly on the progress of the Sprint, and against the Cycle goals, using the [APPROVED] Sprint Standard Reports - Template.

Finishing

Projects **must** finish in the 6-week allocated time so we can extract the benefits of a synchronized planning schedule across the company. In the unfortunate circumstance that projects do *not* finish (either partially or not at all), we have two options:

1. Engineers on the late project can use their "engineering time" to finish delivery.
    a. Pros: creates natural incentives for engineers to finish on time, otherwise they lose their reserved time
    b. Cons: this creates uncertainty for engineering investment execution, since EMs won't know ahead of time if they will have engineers available.
2. We can pause work and kick things back into the Betting Table.
    a. Pros: creates a very predictable schedule on when engineers free up.
    b. Cons:

Risks

Like most processes, this process won't work if we don't stick to it. Doxel has a historical tendency to make last-minute changes to put out fires blowing up. This is a company-level problem that can only be solved if we coordinate Marketing, Sales, Product, and Engineering by clearly defining *what* the product can / cannot do, selling it accordingly, and beating expectations of customers in Pilots and Deployments - so we don't need to scramble to deal with unhappy customers.

To make this process work we'll need buy-in from Robin Singh and Saurabh Ladha that Cycles will not be interrupted, the Phase Gates will not be skipped, etc. This conversation has started already and this execution framework needs to be approved by both of them to ensure that we have alignment across the company.

There are two other major risks:

1. Product may not be "ahead enough" for us to execute this way. As Stan Singh rightly points out, for every Cycle we need a Cycle of Product work to prepare, and we may not have enough Designers and Product managers currently for this to be true. This is a risk that Reid Senescu needs to comment on.
2. We have existing customers/pilots that have been sold products we aren't really planning on continuing (e.g. EAC). Those will always be interruption-driven work and may simply make it impossible for us to have devoted Cycles to new work. We might be able to solve this with air-cover from CS or Sales. This will be a conversation with Saurabh Ladha before we make the decision.

Timeline

2022Q1 has already started so we are going to do an *abbreviated* version of the planning process for our 1st Cycle.

- 2022.01.07: this document approved.
- 2022.01.10: "best-effort" Cycle #1 starts.
- 2022.01.21: Q1 product goals + Engineering roadmap finalized.
- 2020.02.10: Cycle #1 ends.
- 2022.02.10: Pitch docs due, first prioritization meeting.
- 2022.02.17 Second prioritization meeting. Decide on what is going into the Cycle
- 2022.02.24: Start Cycle #2
- 2022.04.07: Cycle #2 ends. Betting Table #3.
- 2022.04.21: Cycle #3 starts.

Appendix

https://basecamp.com/shapeup/webbook
https://buffer.com/resources/6-week-cycles/
https://blog.crisp.se/2016/01/25/henrikkniberg/making-sense-of-mvp
https://www.intercom.com/blog/6-week-cycle-for-product-teams/
https://m.signalvnoise.com/how-we-structure-our-work-and-teams-at-basecamp/#.hstqh0tul
https://www.whitesmith.co/blog/our-framework-to-build-products/