Partner Summit | 2024

Duda Engineering

Be Prepared For The Mountains And Beyond

START JOURNEY
Like most startups, org structure was never really important to us...
Until it became a problem ...
When Duda started scaling, these became increasingly harder for us:

- Communication
- Knowledge sharing
- Decision making
The biggest problem with org structures is that with every org structure, you optimize one aspect at the expense of another...
The first rule of organizational design is that all organizational designs are bad”

BEN HOROWITZ
Here’s the evolution we had...
The early days -
The Jack of all trades

- No real need for a structure
- Everything is very flat
- Everybody does a bit of everything - Code, QA, Devops, Product, UI / UX.

Good times.
The early days

What worked well

- Engagement is through the roof
- Zero friction / communication overhead (No handoffs)
- Knowledge flows freely
- Things are moving FAST
It all worked really well!
Until we started growing...
The early days

**What didn’t work?**

- Onboarding became harder
- Learning curve kept getting steeper
- No clear boundaries and dev processes
- Quality and speed started to drop

*Time for change...*
Teams of Specialists

- Backend
- Frontend
- DevOps
- QA
- Automation
- Product
Teams of Specialists

How did it work

- Each team had a manager (Team Lead)
- We added some processes (Quality Gate, Product Review)
- Projects were done with ad-hoc “task forces”
- Feature Owners made sure things keep ticking
Teams of Specialists

What worked well

- Easy to manage as we had one team per speciality
- High level of mastery and expertise
- Easy for teams to align on guidelines, best practices, code style, etc...
- Hiring process was simple
Teams of Specialists
What didn’t?

- Late and costly integration issues
- Dependency management, cross-team planning, tracking, communication overhead (many slack channels / dailies / syncs)
- Territorialism / friction - “The client should do it!”、“It's clearly the backend's responsibility”
- Lack of focus for teams and TLs
- Scaling further became hard

*Time for change...*
Formed 3 fullstack teams (backend + frontend)

Teams were generic and had no specific domain

Projects were owned by a single team

Teams were paired with a dedicated PM / QA / UX
Fullstack teams

What worked well

- Very flexible - Each team can take on anything
- Communication overhead went down - No cross-teams integrations and less handovers (but still too many)
- Teams ownership went up
Fullstack teams

**What didn’t**

- Not much, worked very well for a long time
- Backlog / Roadmap is centric and hard to scale
- R&D Management became busy with day to day execution, tactics and planning

*Time for change...*
Changing the culture, not (just) the structure
How the new Org should look like

- Scaleable and decentralized org
- Focuses on **outcomes** rather than **outputs**
- Strong ownership by bottom up decision making
- Data driven, not authority / opinion driven
- Remove boundaries and reduce handoffs
- Empower, motivate and inspire our employees
Cross Functional Teams ("Spotify Squad Model")

1. Cross-functional, self reliant, teams that delivers E2E (QA, Devs, PM, UX)
2. Autonomy - “What to build, how to build it, how to work together while doing it”
3. Lean startup principles
4. Boundaries - Long term squad mission
How we implemented it

- 7 squads (7-10 people)
- Squads are a part of a Group with the same business focus
- Squads have a long term, high level mission
- Squad choose their own methodology (Scrum / Kanban / Shape up)
- Each squad has 3 leaders: Team Lead, PM, UX
- Guilds for each discipline (QA, UX, Frontend, Backend, Automation, etc.)
Guilds

- Onboarding new hires
- Knowledge sharing
- Standardization and best practices
- Cross-company initiatives
- Support Hiring
The Squad Model

**Squad**
Main working unit

**Group**
Collection of squads with similar business focus

**Guild**
Group of people with the same set of skills
How do you *really* enable autonomy?

- Management as enablers instead of shots callers
- Provide a tolerant, fail safe environment that supports experimentation
- Allow squads to be truly independent and agile
The design of the software will mirror the design of the organization that built it.

CONWAY’S LAW
Duda’s Monolith

- Editor
- Runtime
- Widgets
- Blog
- Payments
- User Mgmt
- Personalization
- eCommerce
- Data Binding
- Content Library
- Collections
- Widget Builder
Heavy investment in distributed architecture, devops, automation, and code reusability

- Built easy-to-use microservices infrastructure based on common libraries and a rich UI design system
- Fully automated monolith deployments and deploy it 4 times a day
- Aggressively promote trunk-based development and test automation instead of manual regression testing
- Introduced a robust feature flag system
Full CI/CD (Continuous Integration / Delivery)
Green / Blue deployments
Duda's Microservices Architecture

- Store Operations
- Store Catalog
- Cache Server
- Auth Service
- Client Billing
- Dashboards
- App Store
- Duda monolith
- Collections
- Site Optimization
- Analytics
- Custom Branding
- Membership
- Site Comments
- AI Assistant
- Simple Editor
- SSR Widgets
- API Gateway
- Payment Proxy
- Webhooks
Everything is automated without manual QA?
DORA & Google Cloud reports

- The DevOps Research and Assessment (DORA)
- Analyzed and studied over 2000 companies
- Released “Accelerate” in 2018
- Perform long research to understand the relationship between ways of working (processes and capabilities) and outcomes (achievements).
“We see continued evidence that software speed, stability, and availability contribute to organizational performance (including profitability, productivity, and customer satisfaction)”
Technical Key Findings

- Automating code deployment improves software delivery performance and quality
- Trunk-based development boosts performance
- Efficient code review processes contribute to faster software delivery
- Loose coupling is key for independent development, testing, and deployment of services
- Identified 4 key metrics measuring software delivery performance
Speed of Delivery != Reduced Quality
CI/CD unlocks Organizational Success
4 metrics to measure delivery performance

- Lead Time (first commit to production)
- Change failure rate
- Deployment Frequency
- MTTR (mean time to restore)
<table>
<thead>
<tr>
<th>Performance level</th>
<th>Deployment Frequency</th>
<th>Change Lead Time</th>
<th>Change failure rate</th>
<th>Failed deployment recovery time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elite</td>
<td>On demand</td>
<td>Less than one day</td>
<td>5%</td>
<td>Less than one hour</td>
</tr>
<tr>
<td>High</td>
<td>Between once per day and once per week</td>
<td>Between one day and one week</td>
<td>10%</td>
<td>Less than one day</td>
</tr>
<tr>
<td>Medium</td>
<td>Between once per week and once per month</td>
<td>Between one week and one month</td>
<td>15%</td>
<td>Between one day and one week</td>
</tr>
<tr>
<td>Low</td>
<td>Between once per week and once per month</td>
<td>Between one week and one month</td>
<td>64%</td>
<td>Between one month and six months</td>
</tr>
</tbody>
</table>
How does Duda rank?

**Performance level** | **Deployment Frequency**
---|---
Elite | On demand
High | Between once per day and once per week
Medium | Between once per week and once per month
Low | Between once per week and once per month

**Deploy Frequency**

6.24 per day
How does Duda rank?

<table>
<thead>
<tr>
<th>Performance level</th>
<th>Change Lead Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elite</td>
<td>Less than one day</td>
</tr>
<tr>
<td>High</td>
<td>Between one day and one week</td>
</tr>
<tr>
<td>Medium</td>
<td>Between one week and one month</td>
</tr>
<tr>
<td>Low</td>
<td>Between one week and one month</td>
</tr>
</tbody>
</table>

**Cycle Time**

- **1d, 1h** 75th percentile
Failures and Time To Restore

- Blue/Green deployment with automatic rollbacks makes zero-time fixes possible without any manual operations.
- Feature flags provide easy kill switches without code changes and reduced impact when combined with gradual rollouts.
- CI/CD and hotfix mechanism allows shipping code fixes within less than 30 mins.
Failures and Time To Restore - Past 12 months

<table>
<thead>
<tr>
<th>Downtime</th>
<th>Outages</th>
<th>Uptime</th>
<th>Max resp. time</th>
<th>Min resp. time</th>
<th>Avg resp. time</th>
</tr>
</thead>
<tbody>
<tr>
<td>5m</td>
<td>2</td>
<td>&gt;99.99%</td>
<td>508ms</td>
<td>369ms</td>
<td>414ms</td>
</tr>
</tbody>
</table>

Graph showing response time from June 2023 to May 2024, with average response time, downtime, and unknown events.
How can we still get better?
Why even measure?

- “If you can’t measure it, you can’t improve it”
  - Peter Drucker

- Encourages positive behaviour
  - Quick code reviews
  - Commit smaller changes
Since we started measuring ... 

- Improve Cycle Time by 40%
- Reduce code review time by 60%
- Increase deploy freq by 300%

**Deploy Frequency**

173.61 per month

**Review Time**

23h, 33m
What we don’t do

- Get obsessed with scrum metrics like counting story points and team velocity
- Measure developer outputs like lines of code, number of bug fixes, or number of pull requests
- Treat our roadmap as a hard commitment without room to gamble on opportunities, trends, and customer feedback
Pitfalls

- We can still ship features you’ll hate
- We can miss all our business targets and “do well” on DORA
- We are still vulnerable to Goodhart’s law
When a measure becomes a target it ceases to be a good measure

Goodhart’s Law
GOODHART’S LAW

WHEN A MEASURE BECOMES A TARGET,
IT CEASES TO BE A GOOD MEASURE.

If you measure people on...

Then you might get...

Number of nails made

1000's of tiny nails

Weight of nails made

A few giant, heavy nails
Measuring our engineering org

- Measure the development **process** (DORA metrics)
- Combine OKRs/KPIs to measure **business outcomes**
  - Feature Adoption
  - Revenue and growth
  - Churn
  - Customer Satisfaction
Our Philosophy - Summary

- Cross functional squads who deliver E2E and build their own roadmap
- Distributed architecture that supports independent work, safer releases, and quicker cycle times
- CI/CD by investing in automation and devops
- Measuring dev process as a proxy to success and a way to encourage positive behavior
- Measuring business OKRs to make sure we’re doing the right things
What’s new in Core Web Vitals
INP is officially a new metric

FID out, INP kicks in!
Duda keeps improving...

71% of Duda’s sites pass CWV.
Notable recent improvements

- Optimized INP in popups and several other widgets
- Introduced font fallback to eliminate CLS when fonts swapped
- Optimized CLS in mobile sticky header
- Improved critical css stability

This is a long text written in Pacifico, which is a very very very nice font!
What was done in Editor 2.0?

- Reduced tons of unused CSS and JS (500,000 kb reduced!)
- Deprecated old layouts based on client side rendering
- Grid system is pure native web technology (CSS Grid + Flexbox)
- Rewrote pop-ups, drawers, hamburgers, headers, and other basic functionality in a more performant way
Early Editor 2.0 results

- 10-20% better INP
- 10-15% better LCP
- Nearly perfect average of 0.01 CLS!!
- +20-30 points on Lighthouse compared to classic
Thank you.