The Influential Architect
Succeeding At Scale Among Fully Autonomous Teams
Sebastian von Conrad
We’re a marketplace for digital creatives.
Founded in 2006.
Envato

• Founded in 2006.

• Paid > $500m USD to our content authors.
Envato

- Founded in 2006.
- Paid > $500m USD to our content authors.
- > 300 staff worldwide.
Envato

- Founded in 2006.
- Paid > $500m USD to our content authors.
- > 300 staff worldwide.
- > 100 people in the Technology team, reporting to CTO.
Envato

- Founded in 2006.

- Paid > $500m USD to our content authors.

- > 300 staff worldwide.

- > 100 people in the Technology team, reporting to CTO.

- 1 Software Architect. (Me!)
Envato Org Chart

CTO

Senior Eng. Mgr.
Senior Eng. Mgr.
Senior Eng. Mgr.
Senior Eng. Mgr.
Software Architect

The Influential Architect
© 2017 Sebastian von Conrad
Envato

- 4 Groups with 3-8 Software Engineering teams each.
Envato

- 4 Groups with 3-8 Software Engineering teams each.
- 23 Software Engineering teams in total:
Envato

- 4 Groups with 3-8 Software Engineering teams each.
- 23 Software Engineering teams in total:
  - Every team is cross-functional (PO, UX, Frontend, Backend, DevOps.)
Envato

- 4 Groups with 3-8 Software Engineering teams each.

- 23 Software Engineering teams in total:
  - Every team is cross-functional (PO, UX, Frontend, Backend, DevOps.)
  - Every team has 4-8 Engineers.
Envato

‣ 4 Groups with 3-8 Software Engineering teams each.

‣ 23 Software Engineering teams in total:
  › Every team is cross-functional (PO, UX, Frontend, Backend, DevOps.)
  › Every team has 4-8 Engineers.
  › Every team has a Engineering Team Lead.
Envato

- 4 Groups with 3-8 Software Engineering teams each.

- 23 Software Engineering teams in total:
  - Every team is cross-functional (PO, UX, Frontend, Backend, DevOps.)
  - Every team has 4-8 Engineers.
  - Every team has a Engineering Team Lead.
  - Every team has a Technical Lead (usually Team Lead or Senior Engineer.)
Envato

• 4 Groups with 3-8 Software Engineering teams each.
• 23 Software Engineering teams in total:
  • Every team is cross-functional (PO, UX, Frontend, Backend, DevOps.)
  • Every team has 4-8 Engineers.
  • Every team has a Engineering Team Lead.
  • Every team has a Technical Lead (usually Team Lead or Senior Engineer.)
  • One job: answering “why did you do it that way?”
Envato

- 4 Groups with 3-8 Software Engineering teams each.

- 23 Software Engineering teams in total:
  - Every team is cross-functional (PO, UX, Frontend, Backend, DevOps.)
  - Every team has 4-8 Engineers.
  - Every team has a Engineering Team Lead.
  - Every team has a Technical Lead (usually Team Lead or Senior Engineer.)
    - One job: answering “why did you do it that way?”

- 1 Software Architect. (Me!)
Envato

- No central control and command over technology decisions.
Envato

- No central control and command over technology decisions.
- Each team is autonomous and make their own decisions.
Envato

‣ No central control and command over technology decisions.

‣ Each team is autonomous and make their own decisions.
  • Including technologies (languages, databases, etc).
Envato

- No central control and command over technology decisions.
- Each team is autonomous and make their own decisions.
  - Including technologies (languages, databases, etc).
- Each team is accountable for their own decisions.
Envato

- No central control and command over technology decisions.
- Each team is autonomous and make their own decisions.
  - Including technologies (languages, databases, etc).
- Each team is accountable for their own decisions.
- Each team owns systems.
Envato

‣ No central control and command over technology decisions.

‣ Each team is autonomous and make their own decisions.
  ‣ Including technologies (languages, databases, etc).

‣ Each team is accountable for their own decisions.

‣ Each team owns systems.
  ‣ Includes operational responsibility, budgets, etc.
Envato

‣ No central control and command over technology decisions.

‣ Each team is autonomous and make their own decisions.
  ▶ Including technologies (languages, databases, etc).

‣ Each team is accountable for their own decisions.

‣ Each team owns systems.
  ▶ Includes operational responsibility, budgets, etc.

‣ Every system must be owned by a team.
My job is to improve the quality of technical decisions. Without any authority.
More often than not, it feels like my job is to fight entropy.
And I don’t scale very well.
The only way I can affect meaningful change is to influence the decision-makers.
I have to focus on **people**, not technology.
And get them to do my job for me.

Seriously.
Scaling Agile @ Spotify
with Tribes, Squads, Chapters & Guilds

Henrik Kniberg & Anders Ivarsson
Oct 2012
We have an Architecture Guild
Envato’s Architecture Guild
Envato’s Architecture Guild

- It’s a group that discusses and determines things (we'll get to what later).
Envato’s Architecture Guild

› It’s a group that discusses and determines things (we'll get to what later).

› It's a sounding board for technical decisions.
Envato’s Architecture Guild

- It’s a group that discusses and determines things (we'll get to what later).
- It's a sounding board for technical decisions.
- It's a forum for share knowledge and learn from each other.
Envato’s Architecture Guild

‣ It’s a group that discusses and determines things (we'll get to what later).

‣ It's a sounding board for technical decisions.

‣ It's a forum for share knowledge and learn from each other.

‣ Anyone in the Technology team is welcome to participate.
Envato’s Architecture Guild

- It’s a group that discusses and determines things (we'll get to what later).
- It's a sounding board for technical decisions.
- It's a forum for share knowledge and learn from each other.
- Anyone in the Technology team is welcome to participate.
  - Some participants are mandatory. (I get to decide who.)
Let’s talk about the things that we do
Agenda
Agenda

‣ Transparent Decision-making
Agenda

‣ Transparent Decision-making

‣ The Light on the Hill
Agenda

‣ Transparent Decision-making

‣ The Light on the Hill

‣ Architectural Principles
Agenda

‣ Transparent Decision-making
‣ The Light on the Hill
‣ Architectural Principles
‣ Sensible Defaults
Agenda

- Transparent Decision-making
- The Light on the Hill
- Architectural Principles
- Sensible Defaults
- Strategic Design
Disclaimer:
the plural of anecdote is not data.
Disclaimer:
I don’t claim to have come up with anything new.
Transparent Decision-making
Document and Communicate Intentions.
Project docs.
Project Docs

- 2-6 pages.
Project Docs

- 2-6 pages.

- Table of Contents:
Project Docs

- 2-6 pages.

- Table of Contents:
  - What’s this about?
Project Docs

- 2-6 pages.

- Table of Contents:
  - What’s this about?
  - Key constraints
Project Docs

- 2-6 pages.

- Table of Contents:
  - What’s this about?
  - Key constraints
  - Quality Attributes we’re optimising for
Project Docs

- 2-6 pages.

- Table of Contents:
  - What’s this about?
  - Key constraints
  - Quality Attributes we’re optimising for
  - Quality Attributes we’re not optimising for
Project Docs

- 2-6 pages.

- Table of Contents:
  - What’s this about?
  - Key constraints
  - Quality Attributes we’re optimising for
  - Quality Attributes we’re not optimising for
  - Solution summary
Project Docs

- 2-6 pages.
- Table of Contents:
  - What’s this about?
  - Key constraints
  - Quality Attributes we’re optimising for
  - Quality Attributes we’re not optimising for
  - Solution summary
  - Effects on existing systems
Project Docs

- Doesn’t take long to create—a few hours.
Project Docs

- Doesn’t take long to create—a few hours.
- Timing is everything.
Project Docs

› Doesn’t take long to create—a few hours.

› Timing is everything.

› “Ghost writing” services offered.
Project Docs

- Doesn’t take long to create—a few hours.
- Timing is everything.
- “Ghost writing” services offered.
  - Interview team, write for them.
Architect as Bottleneck
Architect as Bottleneck

• “Ghost writing” doesn’t scale.
Architect as Bottleneck

‣ “Ghost writing” doesn’t scale.

‣ Team Lead: “I feel like you’re more ‘ghost’ than ‘writer’.”
Architect as Bottleneck

‣ “Ghost writing” doesn’t scale.
  • Team Lead: “I feel like you’re more ‘ghost’ than ‘writer’.”
‣ Doesn’t have to scale.
Architect as Bottleneck

‣ “Ghost writing” doesn’t scale.
  • Team Lead: “I feel like you’re more ‘ghost’ than ‘writer’.”

‣ Doesn’t have to scale.
  • Establish precedent.
Architect as Bottleneck

‣ “Ghost writing” doesn’t scale.
  ◦ Team Lead: “I feel like you’re more ‘ghost’ than ‘writer’.”

‣ Doesn’t have to scale.
  ◦ Establish precedent.
  ◦ Create templates.
Architect as Bottleneck

‣ “Ghost writing” doesn’t scale.
  ▶ Team Lead: “I feel like you’re more ‘ghost’ than ‘writer’.”

‣ Doesn’t have to scale.
  ▶ Establish precedent.
  ▶ Create templates.
  ▶ Share knowledge
Architect as Bottleneck

‣ “Ghost writing” doesn’t scale.
  • Team Lead: “I feel like you’re more ‘ghost’ than ‘writer’.”

‣ Doesn’t have to scale.
  • Establish precedent.
  • Create templates.
  • Share knowledge
  • Make teams self-sufficient.
Project Docs Feedback
Project Docs Feedback

- Anyone can comment.
Project Docs Feedback

› Anyone can comment.

› *Everyone* comments.
Project Docs Feedback

- Anyone can comment.
  - *Everyone* comments.
- Buy-in from technical stakeholders is desired.
Project Docs Feedback

- Anyone can comment.
  - *Everyone* comments.
- Buy-in from technical stakeholders is desired.
- Revisions are expected.
Project Docs Feedback

- Anyone can comment.
  - *Everyone* comments.
- Buy-in from technical stakeholders is desired.
- Revisions are expected.
- Quick iteration.
Project Docs Feedback

• Anyone can comment.
  
  • *Everyone* comments.

• Buy-in from technical stakeholders is desired.

• Revisions are expected.

• Quick iteration.
  
  • If you miss your chance to comment, tough.
Project Docs Feedback

- Anyone can comment.
  - *Everyone* comments.
- Buy-in from technical stakeholders is desired.
- Revisions are expected.
- Quick iteration.
  - If you miss your chance to comment, tough.
- Tool: Google Docs.
There's a cyclic dependency here between the "Elements Contributor" and "Elements Earnings" systems as represented by this diagram. Discuss :)
I'm nervous about this one. It often goes very much against Affordability. And do the Studio Devs have to be considered as stakeholders?

Perhaps what we're looking for is something that can work on all Market sites and we try to not paint ourselves into a corner for possible future sites?

+1 this sounds like an over-generalisation. At least Studio already has its own system and Elements isn't doing A/B tests yes.
I may be somewhat biased, but I would encourage you to think more broadly about the place of this theme warehouse within our ecosystem - Elements also has themes in it, and the current integration with ThemeForest is less than ideal.

While I wouldn't want you to get derailed trying to build something for both systems, it may be worth considering in detail the particular set of properties that make an item "a theme" - those which are universal between Elements, Hosted and Market?

Thanks for that feedback, the idea of a Theme or Hosted Theme is certainly tricky. The arrival pipeline for a Theme (not just WordPress) is well
I'm curious about the design of the SQS queue's, specifically if you are using Dead Letter Queues and the retention time. But that's an implementation detail.

Thanks. We'll keep you in the loop.
What kind of data are you envisioning to require about an author? I ask as in Content we have the beginnings of a 'source of truth of author information' system starting up and would like some insight into the kinds of data other systems are interested in consuming and whether they would be something we would eventually want to store and provide.

Great question, the first thing that came to mind was badges and all that jazz so when we show an author name along side a theme, we could include some of their fancy details. But I don't know for sure.
Side note: we’re also experimenting with ADRs, go watch Michael Keeling’s talk.
The Light on The Hill
Teams need a **Technical Vision.**
...it’s *not* a roadmap!
Light on the Hill
Light on the Hill

▷ A vision, not a roadmap.
Light on the Hill

› A vision, not a roadmap.

› Our *best guess* of what the future looks like.
Light on the Hill

- A vision, not a roadmap.
- Our *best guess* of what the future looks like.
- Guides technical decision-making.
Light on the Hill

- A vision, not a roadmap.
- Our *best guess* of what the future looks like.
- Guides technical decision-making.
- Keeps teams honest: are we heading in the right direction?
Light on the Hill

‣ A vision, not a roadmap.

‣ Our *best guess* of what the future looks like.

‣ Guides technical decision-making.

‣ Keeps teams honest: are we heading in the right direction?

  ‣ If not (which is okay), at least we’re conscious of it and do it for the right reasons.
Light on the Hill

‣ A vision, not a roadmap.

‣ Our *best guess* of what the future looks like.

‣ Guides technical decision-making.

‣ Keeps teams honest: are we heading in the right direction?
  ‣ If not (which is okay), at least we’re conscious of it and do it for the right reasons.

‣ Each group has their own Light on the Hill.
Light on the Hill

- A vision, not a roadmap.
- Our *best guess* of what the future looks like.
- Guides technical decision-making.
- Keeps teams honest: are we heading in the right direction?
  - If not (which is okay), at least we’re conscious of it and do it for the right reasons.
- Each group has their own Light on the Hill.
  - The group’s Senior Engineering Manager (or their designated delegates) owns the Light on the Hill.
Great results, through business and product stakeholder buy-in.
Expressing a Light on the Hill
Expressing a Light on the Hill

- Initially, a big giant doc/diagram for each group.
Expressing a Light on the Hill

- Initially, a big giant doc/diagram for each group.
- Yeah, that didn’t work.
Expressing a Light on the Hill

‣ Initially, a big giant doc/diagram for each group.

‣ Yeah, that didn’t work.

‣ Hard to maintain: got out of date quickly.
Expressing a Light on the Hill

- Initially, a big giant doc/diagram for each group.
- Yeah, that didn’t work.
  - Hard to maintain: got out of date quickly.
  - Information overload.
Expressing a Light on the Hill

› Initially, a big giant doc/diagram for each group.

› Yeah, that didn’t work.

   › Hard to maintain: got out of date quickly.

   › Information overload.

   › Difficult for technical folks, much less non-technical stakeholders, to wrap their heads around it.
Instead, we sliced by capability.
...and used visual storytelling.
The Anatomy of a Story
The Anatomy of a Story

› In the past...
The Anatomy of a Story

- In the past…
- Then something happened…
The Anatomy of a Story

‣ In the past…

‣ Then something happened…

‣ Now we’re…
The Anatomy of a Story

‣ In the past…

‣ Then something happened…

‣ Now we’re…

‣ So that in the future…
Let’s use a real example!
Federated User Information

Light on the Hill
In the beginning, products like Market had all information about their users in their own databases.
Then when we were building Studio, we decided that we would benefit from sharing user information between products.

We built account.envato.com (SSO) to provide a unified sign up/in experience and share user information between products.
Of course, not all information made its way over to SSO. Both Market and Studio also collected their own information about users.
Over time, more of the products we built were integrated with SSO, all the while also capturing their own information relevant to them.
We also wanted to share more information than just basic user details, such as billing details and tax information.

We built a system called **Identity** to hold this information.

Now, to get the information about users product wanted, they had to integrate with both Identity and SSO.
We are now finding that there is information in the products themselves that other products are interested in.

For example, Market might want to know if a user is an Elements subscriber so they can give discounts on Market purchases.
What we’ve ended up with is that different products and systems each hold a different aspect of what we know about a user.

Some of the systems that have key information are ones we haven’t built ourselves, like Zendesk or Discourse (our community forums).
With information about users scattered around different products and systems, giving products and systems the information they need is quickly becoming a problem.

Lots of integration points and interdependencies makes systems harder to manage and change.
In the future, we will build a new system (tentatively called X) which holds a federated view of users.
To get information into X, a **Watcher** will “watch” a particular source of user information.

The Watcher will send the information it retrieves from the source to X.

X will only ever have a *copy* of the information. The source will still be the source of truth.
There will be many Watchers that each watch different systems for information. Some Watchers may watch several systems, while others only watch one.
The many Watchers will push the information they retrieve from the source systems to X.
In collecting this data, X can present a federated view of the information we have about the user.

Over time, products will start asking X for information about users, rather than integrating with every other system directly.
And in the end, the Light on the Hill is for all products and systems to only ever ask X for information about users.
The intended audience is **not** technical folks.
It’s to get broad agreement on technical vision with business and product owners.
This meme has caught on.
Developing a Light on the Hill
Developing a Light on the Hill

• Workshop with selected participants:
Developing a Light on the Hill

‣ Workshop with selected participants:

‣ Key tech and product people from group.
Developing a Light on the Hill

› Workshop with selected participants:
  
  › Key tech and product people from group.
  
  › Stakeholders from other dependent groups.
Developing a Light on the Hill

‣ Workshop with selected participants:
  ‣ Key tech and product people from group.
  ‣ Stakeholders from other dependent groups.
  ‣ Senior stakeholders (e.g. CTO, CFO) if interested.
Developing a Light on the Hill

› Workshop with selected participants:

› Key tech and product people from group.

› Stakeholders from other dependent groups.

› Senior stakeholders (e.g. CTO, CFO) if interested.

› Outcome is a draft.
Developing a Light on the Hill

- Workshop with selected participants:
  - Key tech and product people from group.
  - Stakeholders from other dependent groups.
  - Senior stakeholders (e.g. CTO, CFO) if interested.

- Outcome is a draft.

- Share with Architecture Guild for feedback, similar to Project Docs.
Developing a Light on the Hill

‣ Workshop with selected participants:
  • Key tech and product people from group.
  • Stakeholders from other dependent groups.
  • Senior stakeholders (e.g. CTO, CFO) if interested.

‣ Outcome is a draft.

‣ Share with Architecture Guild for feedback, similar to Project Docs.

‣ Once ratified, share far and wide. (Demos, exec team meetings, etc.)
Developing a Light on the Hill

- Workshop with selected participants:
  - Key tech and product people from group.
  - Stakeholders from other dependent groups.
  - Senior stakeholders (e.g. CTO, CFO) if interested.

- Outcome is a draft.

- Share with Architecture Guild for feedback, similar to Project Docs.

- Once ratified, share far and wide. (Demos, exec team meetings, etc.)

- Refer to them often.
Architectural Principles
Broad, high-level principles we want all teams to follow.
Architectural Principles
Architectural Principles

- $7 \pm 2$. 
Architectural Principles

- $7 \pm 2$.
- Expressed as value statements.
Architectural Principles

▶ 7 ± 2.

▶ Expressed as value statements.

▶ Technology/pattern agnostic.
Architectural Principles

› 7 ± 2.

› Expressed as value statements.

› Technology/pattern agnostic.

› Examples:
Architectural Principles

‣ 7 ± 2.

‣ Expressed as value statements.

‣ Technology/pattern agnostic.

‣ Examples:

  ‣ *We prefer asynchronous communication.*
Architectural Principles

• $7 \pm 2$.

• Expressed as value statements.

• Technology/pattern agnostic.

• Examples:
  
  • *We prefer asynchronous communication.*

  • *We keep dependencies acyclic and stable.*
Architectural Principles

‣ 7 ± 2.

‣ Expressed as value statements.

‣ Technology/pattern agnostic.

‣ Examples:

  › We prefer asynchronous communication.

  › We keep dependencies acyclic and stable.

  › We define Quality Attributes for systems and design with them in mind.
Architectural Principles
Architectural Principles

- They apply to everything technical decision we make.
Architectural Principles

‣ They apply to everything technical decision we make.

‣ They are vehicles for accountability.
Architectural Principles

‣ They apply to everything technical decision we make.

‣ They are vehicles for accountability.

‣ Anyone is able to hold anyone else accountable to them.
Architectural Principles Process
Architectural Principles Process

- Anonymous suggestion box.
Architectural Principles Process

‣ Anonymous suggestion box.

‣ Curation => first draft.
Architectural Principles Process

› Anonymous suggestion box.

› Curation => first draft.

› Deep discussion on each one => Second draft => Third draft.
Architectural Principles Process

• Anonymous suggestion box.

• Curation => first draft.

• Deep discussion on each one => Second draft => Third draft.
  • Buy-in was crucial.
Architectural Principles Process

• Anonymous suggestion box.

• Curation => first draft.

• Deep discussion on each one => Second draft => Third draft.
  
  • Buy-in was crucial.

• Once finalised, published.
Sensible Defaults
We see patterns emerge across teams.
Same problem, different team.
Don’t want to reinvent the wheel.
Sensible Defaults
Sensible Defaults

- Off-the-shelf solutions to common problems.
Sensible Defaults

- Off-the-shelf solutions to common problems.
  - E.g. API Design, Authentication, Logging, 3rd Party Integration, Identifying and Handling Sensitive Data.
Sensible Defaults

‣ Off-the-shelf solutions to common problems.
  ‣ E.g. API Design, Authentication, Logging, 3rd Party Integration, Identifying and Handling Sensitive Data.
  ‣ Typically originates in practice, not theory.
Sensible Defaults

‣ Off-the-shelf solutions to common problems.

‣ E.g. API Design, Authentication, Logging, 3rd Party Integration, Identifying and Handling Sensitive Data.

‣ Typically originates in practice, not theory.

‣ Described as specifications.
Sensible Defaults

‣ Off-the-shelf solutions to common problems.

‣ E.g. API Design, Authentication, Logging, 3rd Party Integration, Identifying and Handling Sensitive Data.

‣ Typically originates in practice, not theory.

‣ Described as specifications.

‣ Specifications frequently lead to libraries.
Sensible Defaults

‣ Off-the-shelf solutions to common problems.

‣ E.g. API Design, Authentication, Logging, 3rd Party Integration, Identifying and Handling Sensitive Data.

‣ Typically originates in practice, not theory.

‣ Described as specifications.

‣ Specifications frequently lead to libraries.

‣ Includes why as well as what, or we may miss an opportunity to teach.
Sensible Defaults

- Off-the-shelf solutions to common problems.
  - E.g. API Design, Authentication, Logging, 3rd Party Integration, Identifying and Handling Sensitive Data.
- Typically originates in practice, not theory.
- Described as specifications.
  - Specifications frequently lead to libraries.
- Includes *why* as well as *what*, or we may miss an opportunity to teach.
  - The *why* also enables the sensible default to be challenged.
Sensible Default example: Authentication

- MUST use request signing.
- MUST sign either the entire request body or a hash of the entire request body.
  - Hashing is RECOMMENDED when the request body is large. Use SHA256 or better for hashing.
- MUST sign the following request headers (if they are sent): Content-Length, Content-Type, Date, User-Agent
- MUST sign the request path.
- MUST set an expiry time for the request.
- SHOULD sign any other request headers you deem appropriate.
- MAY send a unique identifier for the request and validate server-side to prevent replay attacks.
- MUST use TLS/SSL for the request.
- SHOULD obtain an externally verifiable certificate for TLS/SSL.
- SHOULD whitelist IP addresses of expected clients and fail authentication for requests from other IPs.
- MUST drop any requests that failed authentication.
- MUST log any requests that failed authentication.
- MAY raise on-call alerts for requests that failed authentication if warranted.
Sensible Defaults
Sensible Defaults

- Sensible Defaults are recommended, but never mandated.
Sensible Defaults

‣ Sensible Defaults are recommended, but never mandated.
  ▷ You’ll never get in trouble for using one.
Sensible Defaults

‣ Sensible Defaults are recommended, but never mandated.
  ‣ You’ll never get in trouble for using one.
  ‣ If you choose not to use one, you’ll be asked why.
Sensible Defaults

‣ Sensible Defaults are recommended, but never mandated.
  ‣ You’ll never get in trouble for using one.
  ‣ If you choose not to use one, you’ll be asked why.
‣ We don’t want to stymie innovation.
Sensible Defaults

✱ Sensible Defaults are recommended, but never mandated.
  ✷ You’ll never get in trouble for using one.
  ✷ If you choose not to use one, you’ll be asked why.

✱ We don’t want to stymie innovation.
  ✷ Have a better idea? Go for it!
Sensible Defaults

‣ Sensible Defaults are recommended, but never mandated.
   ‣ You’ll never get in trouble for using one.
   ‣ If you choose not to use one, you’ll be asked why.

‣ We don’t want to stymie innovation.
   ‣ Have a better idea? Go for it!
   ‣ We can update the default.
Sensible Defaults Process
Sensible Defaults Process

- Anyone can write a Sensible Default.
Sensible Defaults Process

› Anyone can write a Sensible Default.

› Same process as for everything else: open discussion and iteration.
Strategic Design
This may be controversial…
Architecture exists to serve the needs of the business.
A better architecture allows us to go faster and say “yes” more often.
As such, our architectural goals must follow your business goals.
CEO says “I want to do X…”
Here’s the thing: sometimes the shortcut is right.
I often find myself representing the idealism, out of necessity.
The number one trait I look for in Tech Leads is **balance** between Idealism and Pragmatism.
Strategic Design
Strategic Design

- Understand business objectives and strategy.
Strategic Design

- Understand business objectives and strategy.
- Weigh tradeoffs.
Strategic Design

‣ Understand business objectives and strategy.

‣ Weigh tradeoffs.

‣ The design has to be proportionate to the business value.
Strategic Design

‣ Understand business objectives and strategy.

‣ Weigh tradeoffs.

‣ The design has to be proportionate to the business value.

‣ Use the Strategic Design graph.
Strategic Design

- Buy
- KISS
- WTF?

Axes:
- Competitive advantage
- Complexity
In Summary
I’m not gonna lie…
...sometimes my job would be easier if I could just tell people what to do.
…or at least act as a gatekeeper for their decisions.
But with 23 autonomous teams, that doesn’t scale.
So how do I improve the quality of our Technical Decision-making at scale?
What controls do I put in place?
By making teams accountable to each other.
Sometimes the conversations are confronting.
“The Architecture Guild is where you go when your impostor syndrome has been dormant for a while.”

- Envato Tech Lead
“The Architecture Guild is like baring your soul to the world.”

- Envato Senior Engineer
But if I’ve done my job right, the difference is not in the decisions themselves...
…but the people who make them.
They will express their technical vision as **Lights on the Hill.**
They will communicate their intent via **Project Docs**.
They will adhere to Architectural Principles.
They will use (or don't use) **Sensible Defaults.**
They will be **Strategic in their Designs.**
Thank you.