Why changing the way we architect our solutions?

Table of Contents

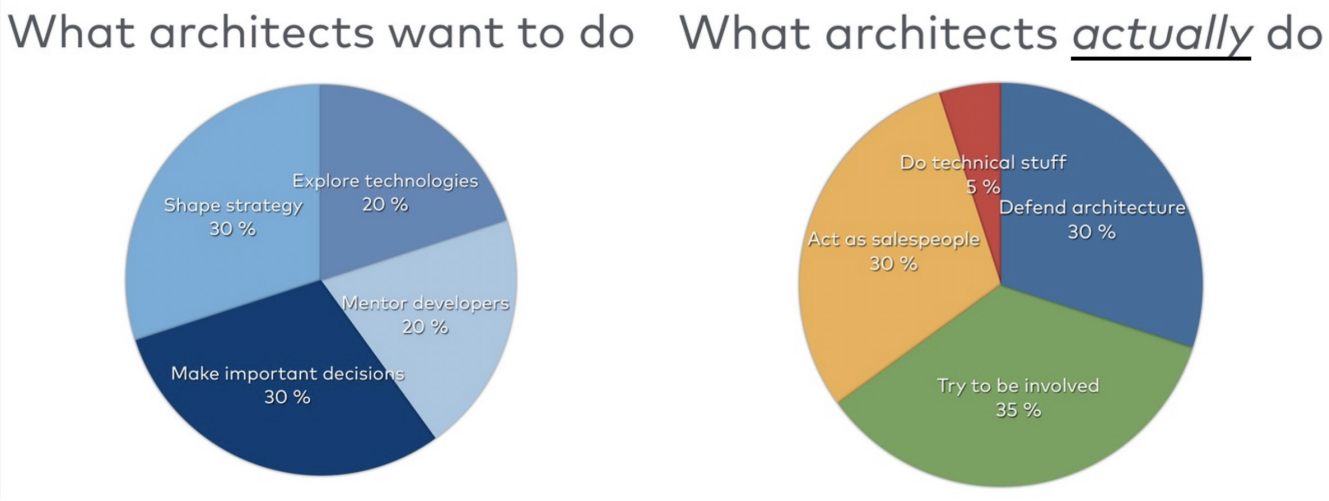
The environment we're living in is often qualified using the VUCA acronym standing for Volatility, Uncertainty, Complexity & Ambiguity. In such context, the only thing we can do is to build a resilient, flexible and reactive open information system. We can't predict what the near future will be so lets design our Information System in a way it can evolve whatever the evolution looks like.

"Yesterday's architecture methodologies and processes will not deliver future solutions.". It's our credo. We need to change our operating model to maintain our system integrity @scale and to make people autonomous and safe when they exercise initiative. We're coming from a world where architecture decisions were taken centrally and we know it'll become an unbearable bottleneck. But decentralisation is more complex than simply delegating authority. In other words, ALIGNMENT + AUTONOMY > CONTROL.

There is another big change we have to deal with: software engineering became continuous. We are continuously building and deploying our solutions to deliver value regularly to our users. As a consequence, we must continuously make important decisions correctly and quickly: that's what we call Continuous Architecture.

1. Rituals

Unless you're practicing architecture every day, we often spend our time on very different activites as depicted on the below charts.



(courtesy from Stefan Tilkov).

So if you find yourself in this situation, we have a ritual that was created to help you: it's the architecture kata and we want to give credits to Ted Neward who formalized it several years ago.

1.1. Why having a ritual to practice architecture ?

The two below quotes are self explanatory we believe:

> "How do we get great designers? Great designers design, of course." --Fred Brooks

> "So how are we supposed to get great architects, if they only get the chance to architect fewer than a half-dozen times in their career?" --Ted Neward

Naming this ritual kata is a direct analogy to the Japanese Kata word (□, □) where it literally means "form referring to a detailed choreographed pattern of martial arts movements made to be practised aloneor within groups and in unison when training. It is practised in Japanese martial arts as a way to memorize and perfect the movements being executed." — Wikipedia

The idea of an architecture kata is then to repeat the architecture "movement" often enough to memorize it and perfect it over time.

1.2. How the ritual is conducted?

There are only two pre-requisites to organize an architecture kata: you need a topic / problem to work on and you need to have at least 2 hours ahead of you.

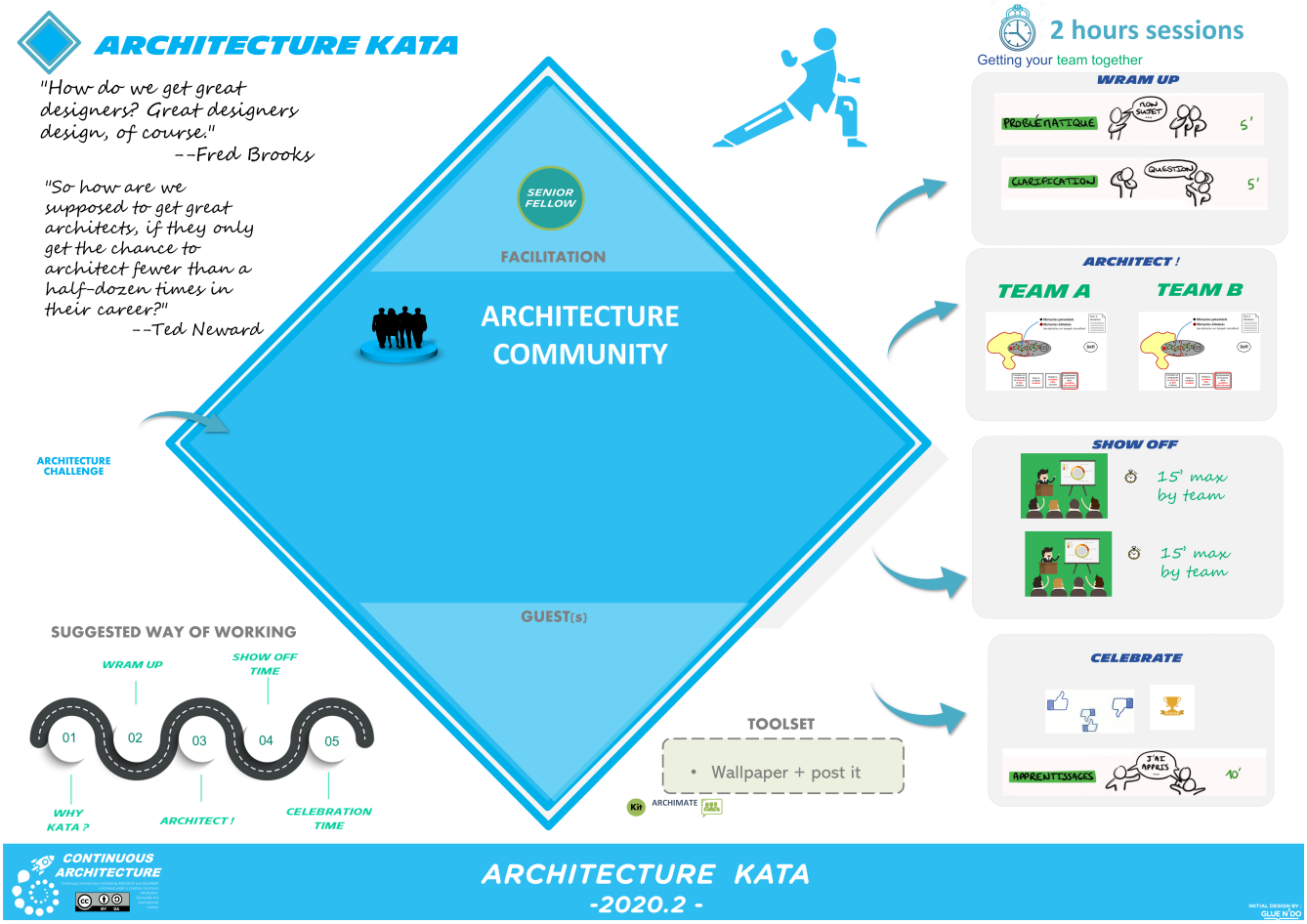
A kata is organized is 4 stages

1. Warm-up phase: the moderator (who can be seen as the customer) presents the topic / problem he wants the different groups to work on. He takes time to answer any clarification questions so attendees discovers the topic. Then we assemble the groups / teams. Depending on the number of persons attending the kata, we can create one or several teams of 5 people. If you have multiple teams, there will be another benefit to this practice: having several teams solving your

problem, you may end up finding one solution or more ;)

2. Architect phase: each team is trying to outline an architecture that fix the given problem. And there are very few rules to respect
    - You may ask the Moderator any questions you have about the topic/problem.
    - Any technology is fair game. Customers really don't care, most of the time, what kind of technology you use. Just be prepared to defend it use during the show off Phase.
    - You may safely make assumptions about technologies you don't know well. However, any assumptions you make under this rule must be clearly defined and described during the show off phase.
    - You may not assume you have hiring/firing authority over the development team. No assumptions of developer All-Stars; assume that a development team roughly as skilled as the one you work with on a daily basis will be doing the implementation.
3. Show off phase (or peer review): During this phase, your team will be either presenting to other groups, or listening to other groups. If you are presenting, you have to explain your proposal and answer questions on it. If you are listening to the other groups, your job is to ask questions and please try to keep the questions constructive. But feel free to openly question any choice or decision that you think might not have been carefully examined or thought out. The moderator here will help to lower down the tension if a kind of back-and-forth passionate discussion starts. Remember that teams may assume anything about a technology they don't know well, so long as that assumption is clearly spelled out; if they assumed something that you know to be false, by all means inform them of that, but bear in mind that someone else in the room may have different experience with it than you, and keep an open mind. Also keep in mind that you'll be presenting your solution soon ;)
4. Vote and celebration phase: after each team has finished their presentation, we move to the voting phase. The Moderator will call out a "1-2-3", and you will each individually give the team an overall feedback vote:
    -  : You thought they nailed it, answered all obvious questions, chosen credible and feasible tech that at least seems credible and feasible, and they have a basic vision of solution. This doesn't mean they have every answer, or that they have already produced a UML diagram, it means that you have confidence they could produce them given enough time.
    -  : You thought they missed a few things, enough to make you a bit uncomfortable that they really have a clear vision of what they're trying to build. They forgot some key questions to ask the customer, they forgot some important aspect, or they just in general didn't give you the feeling that they really "got it".
    -  : You thought they missed it. They made major assumptions that you think had no validity to it. They never asked the customer any questions and got burned on a few bad assumptions as a result. They really bungled the job, and you would never want to work with them on this topic.
    - But all is not finished! Once the voting is complete, the ancient Klingon proverb must be heeded: "Revenge is a dish best served cold". The team departing the stage chooses the next one, and we repeat again by going back to the Vote and celebration Phase for them.

The below visual recaps how an architecture kata is conducted:



One of the key objective of Continuous Architecture operating model is to safely entrust product teams in taking architecture decisions to enable end-to-end ownership and continuous delivery. This means that we need to decentralize decision making around architecture while keeping the integrity of an information system which is usally compose of thousands of applications with a complex interweaving flows of data between them.

To achieve this objective and as different delivery teams require varying levels of oversight to successfully conduct architecture activities, C.A. operating model proposes a formal approach to evaluate their level of readiness for autonomy in taking architecture decisions based on the context. This assessmeent is done on a regular basis, over a 12 to 18 months horizon, during the Team readiness for autonomy ritual and formalize using the tool Team Autonomy readiness poster.

1.3. The team autonomy readiness poster

The tool propose 3 different levels of centralization/decentralization of architecture decisions which are:

- **Level 1**: The delivery team is autonomous to make the architectural decisions necessary for the evolution of the products it manages. Product & Fullstack architects provide support on-demand.
- **Level 2**: Architectural decisions are taken under the guidance of Product and Fullstack architects in charge of the domain/product line to which the product belongs. They lead the

collaboration between all actors, ensure the end to end integrity and consistency of the system at scale while developing delivery teams judgement (coaching & mentoring) to increase their autonomy.

- **Level 3**: Architectural decisions need to be taken centraly under governance of enterprise architecture. EA architects educate architects community and delivery teams on new architecture considerations to increase their autonomy.

The level of readiness is defined delivery team by delivery team based on the assessment around 2 dimensions.

1. On **Y** axis: the delivery team maturity regarding:
    - knowledge and mastery of architecture guiderails (principles, blueprints & standards) of the organisation it belongs to.
    - technical and functional skills assessment of the delivry team's members.
2. on **X** axis: the level of architecture risks related to the product considering among other things:
    - the stage of the product in its life cycle
    - dependencies with other products and cross-team decisions needs
    - security/legal consideration for the company
    - level of investment
    - sourcing that engage the company

[Team Readiness for Autonomy] | *KIT%20Generic%20Autonomy%20Assesment 2020.2.png*

Together, the team's functional & technical maturity and the level of architecture risk determine its level of readiness to self-manage architectural decisions on products it manages.

As one of the responsability or the architect role in C.A. operating model is to educate delivery teams in design and architecture considerations, the assessment ritual is the opportunity to evaluate C.A practices adopted (the A symbol on the poster) and identify areas to improve with the delivery team (T symbol in the poster).

ToDo: voir avec Nicolas comment mieux expliquer la partie droite du poster: La roue progress plan? la zone A/T ?

1.4. The team autonomy readiness assessment ritual

Evaluation of teams autonomy readiness is done on a regular basis during this ritual animated and facilitated by the Product architect and the Fullstack architect for the product or line of product they have in charge of.

[Team Readiness for Autonomy] | *Continuous-architecture%20Generic%20Rituals%20-*

%20Readiness%20for%20Team%20Autonomy%20-%202020-2.png

The participants are usually, the Delivery Manager, the Lead Architect, the EA Architect and optionaly techical leaders and team leaders of delivery squads.

The main outcome of the meeting is obviously the assessement of teams formalized in the Team Autonomy readiness poster but the dialogue between the parties is just as important. Possibly a specific organization and governance can be set up around architecture decisions for the related topic. And the ritual is also the oppotunity to update Continuous Architecture progress plans and competences plans f]r delivery teams and architects as needed.

For preparation, Product/Fullstack architects have to consider following inputs:

- Products delivery roadmap for the next 12 to 18 months
- Last Continuous Architecture progress plans review
- Global architecture asssements of their product portfolio (level of obsolecence & IT Debt, alignment with architecture principles & blueprints, quality of service, quality of experience,...)

ToDo: Demander à Nicolas de relire et compléter ci-besoin.

1.5. Team autonomy readiness assessment outcome example

ToDo: mettre un example ici d'un assessment.