City Building: Iteration 3

Outro

> There are only two hard things in Computer Science: cache invalidation and naming things.
>
> *Phil Karlton*

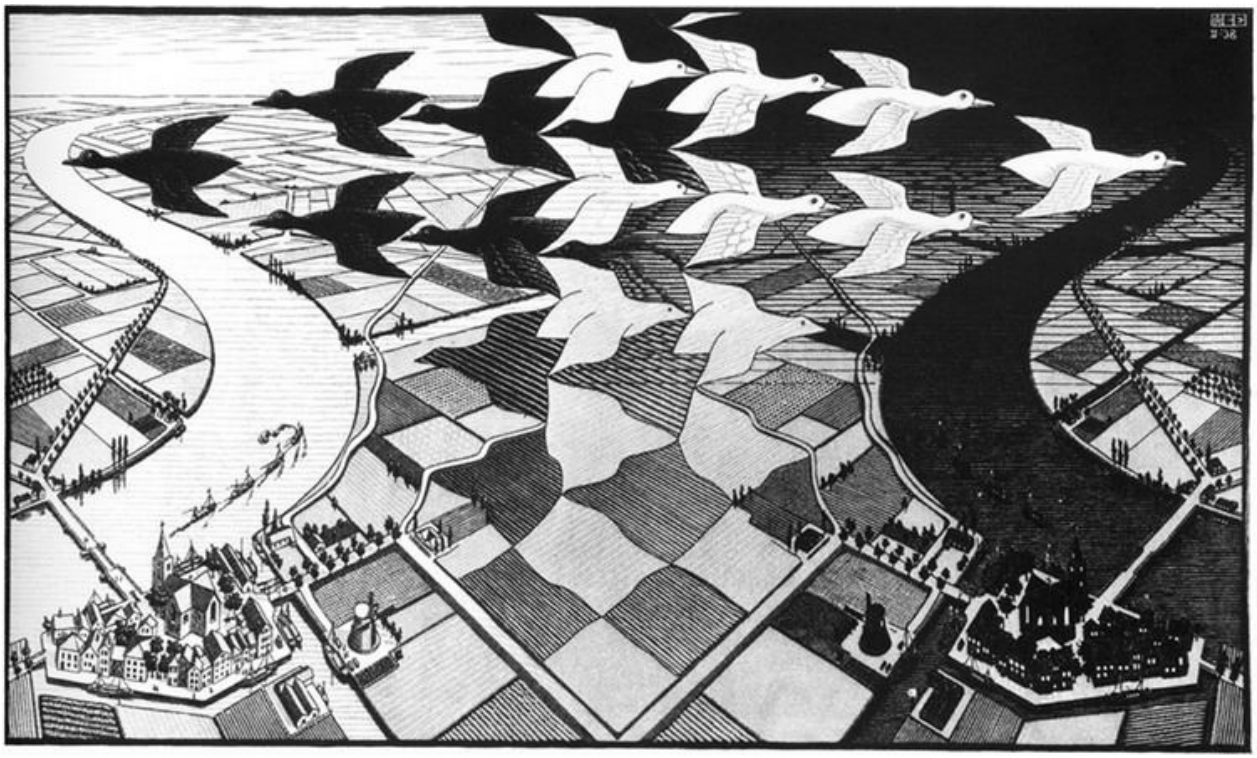


Figure 1: We're nothing more than a *strange loop*

It's a beautiful day and you're daydreaming while listening to your favourite song. The lyrics then suddenly express the words: "The system isn't as scalable as we would like it to be. Is there some way I can blame you for this?" While you're not sure how your boss' utterances have infiltrated the music streaming service, the command has reached your subconscious layer and you have to obey. Good design is hard but luckily you have the required taste to overcome such inane difficulties. You deduce that the backend interaction model requires some rethinking. The big realization is that there are several components with similar responsibilities and interaction rules across the system. This is a powerful idea that improves the meta-model and you decide to abstract the background workers under a single concept: **Agent**. The user always expresses an intent when he/she interacts with the application. As a direct result of that, we would like to map the intents to some internal

action that can be performed by some agent. To make this work, you annotate each agent with two new fields:

- **noun** - the entity that can be processed by the agent
- **verb** - the transformation that can be performed on an entity

Thus, the calling side needs to specify the subject and the predicate of the intent, with an optional parameter which mentions a particular agent that is meant to perform the action.

Given all the registered agents in the application, the system generates an index for faster resolution of the intent. An incoming intent is split into a collection of nouns and verbs. The framework uses the aforementioned index to find the best tool for the job. Upon doing so, it starts the agent and passes the necessary arguments to its internal intent handler - the interface the framework will connect to each time an intent is detected. The framework should be robust enough to recognize if the needed agent is already running. In that case, it should only change the entry point arguments of the agent. There's just one little footnote: our users expect to talk to their computing devices, and we're still using last year's interaction method. We need a way to map natural language commands to the above-mentioned intents; the resolution module should take care of the rest. I suggest you get to work before the media portrays us next to the Great Wall of China, suggesting that we're the only programming failure visible from space.

Requirements and food for thought

1. Implement the agent system and link it to the NLP module.
2. There are multiple ways in which bounded contexts can communicate with each other:
    - A **shared kernel** model is where two contexts share some common domain design, and so you must collaborate.
    - A **customer/supplier** or **consumer driven contract** model is where the downstream context defines the contract that they want the upstream context to provide. The two domains can still evolve independently, as long as the upstream context fulfills its obligations under the contract[1].
    - A **conformist** model is the opposite of consumer driven. The downstream context accepts the contract provided by the upstream context, and adapts its own domain model to match.
3. Use good names. Naming things is hard and also surprisingly important. Since naming is a design tool, many of the principles you already know from writing good code

[1] Service-Oriented Development with Consumer-Driven Contracts

apply to architecture too. Here are Belshee's 7 stages of naming applied to software architecture[2]:

- **Missing** - No name. We don't know enough about the system or context to extract a named element.
- **Nonsense** - Name has absolutely no meaning. We have identified a chunk of ideas as being somehow related.
- **Honest** - The name describes at least one of the element's responsibilities.
- **Honest and Complete** - The name directly describes *all* of the element's responsibilities.
- **Does the Right Thing** - The name reflects a conscious decision to evolve the element's responsibility. This only happens as we gain more knowledge about the element's role in the context of the architecture.
- **Intent** - The name describes the element's responsibility but also its purpose. Understanding purpose requires that we understand *why* the element exists in addition to what it doesn't.
- **Domain Abstraction** - The name transcends individual elements to create a new abstraction. This is where new concepts for the meta-model are born.

Here's an example of one naming progression. In this project, the designers were attempting to name a set of elements responsible for fetching data from a web service and transforming it.

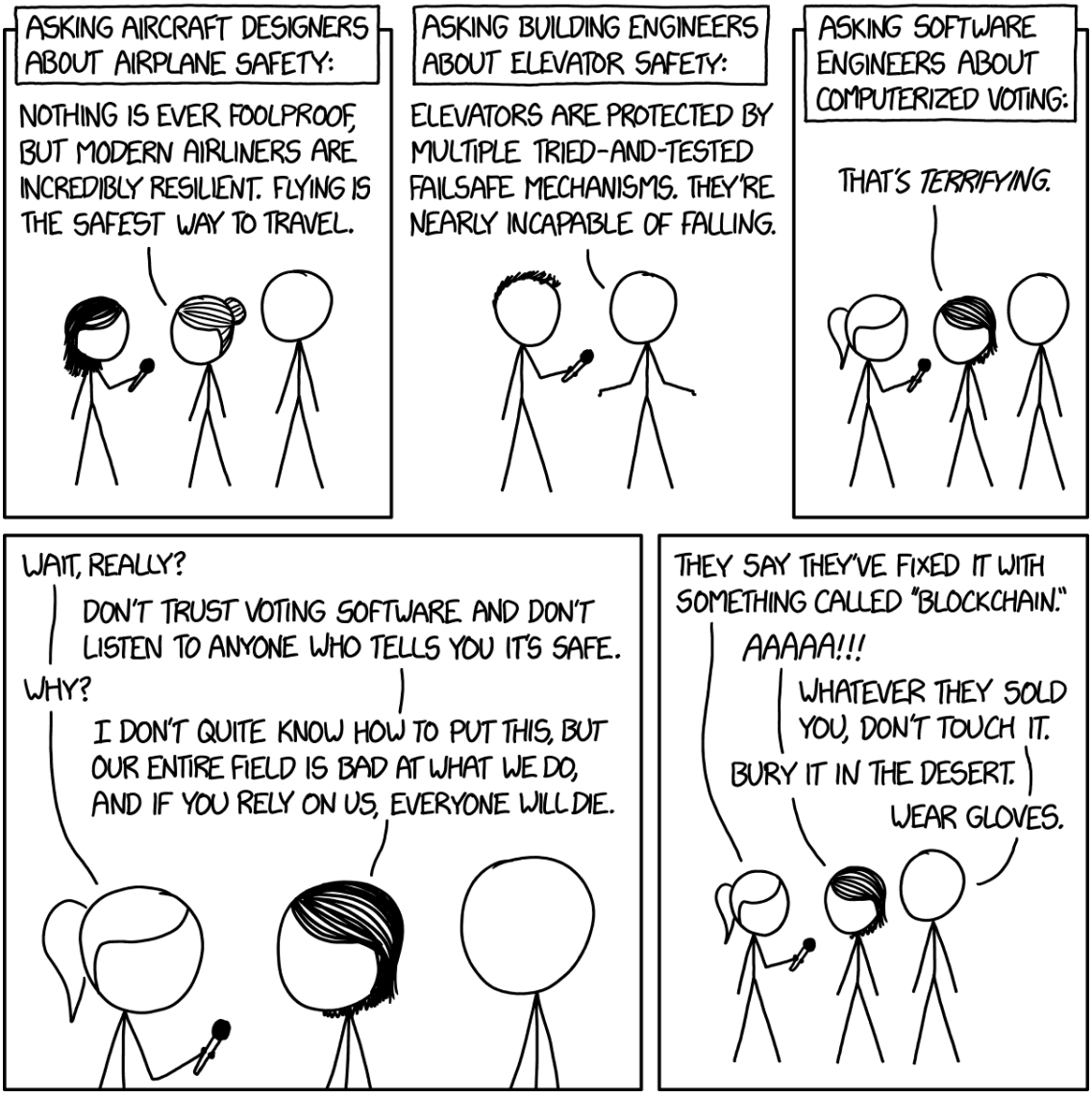
Stage	Name
1. Missing	The thing that does the thing
2. Nonsense	Cranberry
3. Honest	Job Starter Process
4. Honest and Complete	Data Fetcher, Checker, Transformer, and Job Starter
5. Does the Right Thing	Data Transformation Job Runner
6. Intent	Data Preparer
7. Domain Abstraction	Data Preparation Agent

Use names as a litmus test to determine how well you understand the concepts in the architecture. If your names are nonsense or simply honest, then you may have more work until you understand the concepts you're designing[3].

[2] Good naming is a process, not a single step
[3] Design It! by Michael Keeling

City Building: Iteration 3

4. What do you get when you cross an alarm clock with a computer? What do you get when you cross a computer with a car? What do you get when you cross a computer with a warship? What do you get when you cross a computer with an airplane? You get a *computer*, of course. You cannot set an alarm without going through 5 different menu items anymore, a single *NULL* pointer exception on one terminal is enough to strand an entire warship in the middle of the ocean and a computer interface design decision is enough to cause a *human error* that, in turn, causes the death of 159 passengers aboard Flight 965. What do you get when you cross a computer with a city?[4].



[4]The Inmates Are Running the Asylum by Alan Cooper