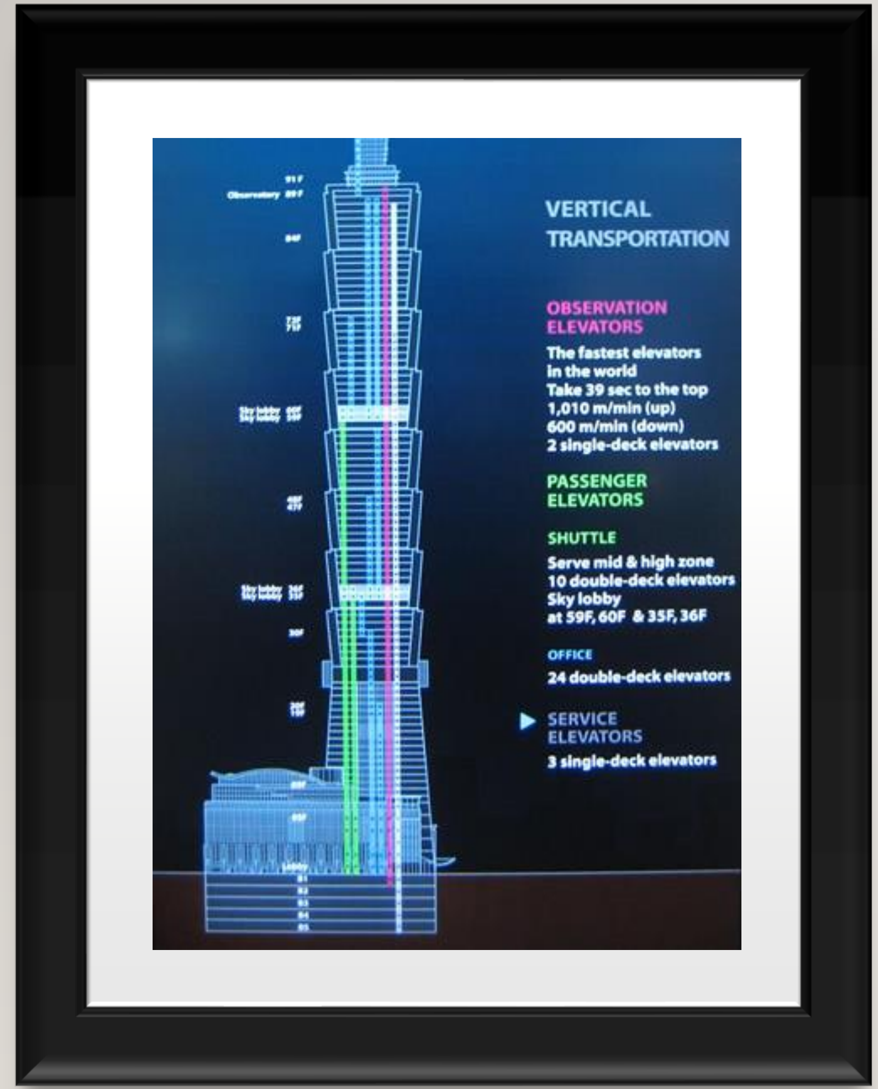
COMMUNICATION



THE ARCHITECT ELEVATOR

"This is what I call the architect elevator: architects ride the elevator up and down to move between the board room and the engine room of a large enterprise."

— Gregor Hohpe

http://www.enterpriseintegrationpatterns.com/ramblings/79_elevator.html

ARCHITECT AS TRANSLATOR

- Technology team
- Program/project management
- Stakeholders
- And posterity

ARCHITECT AS SHAMAN

Developer: "This is stupid! Why is this here?"

Architect: "Ah, come friend. Sit by the fire and let me tell you the story of our system."

— Dan North

ON DOCUMENTS

ON DOCUMENTS

The document is not the work.

It is a record of the work.

“Done with the document” doesn’t mean “done with the work”

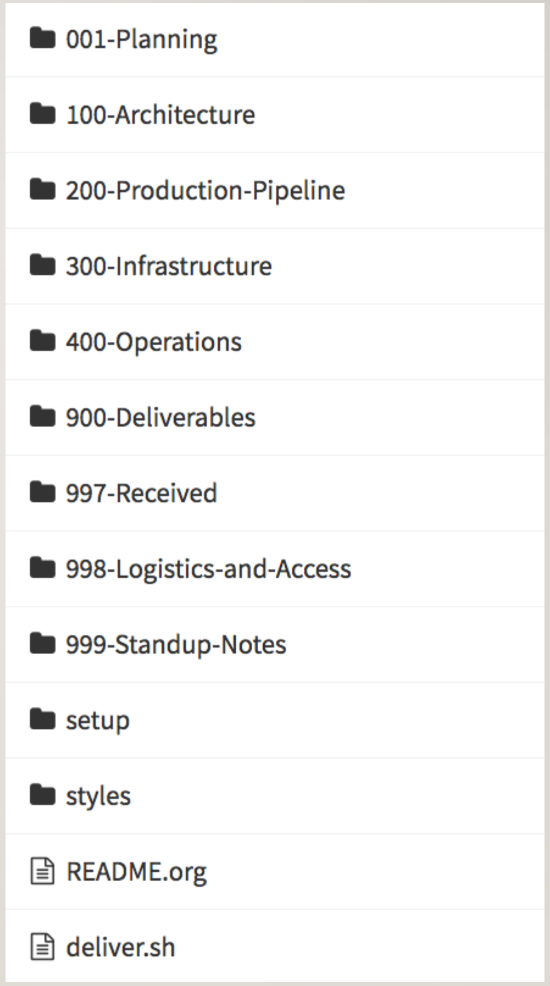
DOCUMENTS CAN

- Capture information
- Present your thinking
- Identify areas to explore

COMMUNICATION TOOLS

ARCHITECTS FOLDER

- Reminders to myself about areas to consider
- Useful scaffolding, but adapted for every project and system
- Mostly text



001-Planning
100-Architecture
200-Production-Pipeline
300-Infrastructure
400-Operations
900-Deliverables
997-Received
998-Logistics-and-Access
999-Standup-Notes
setup
styles
README.org
deliver.sh

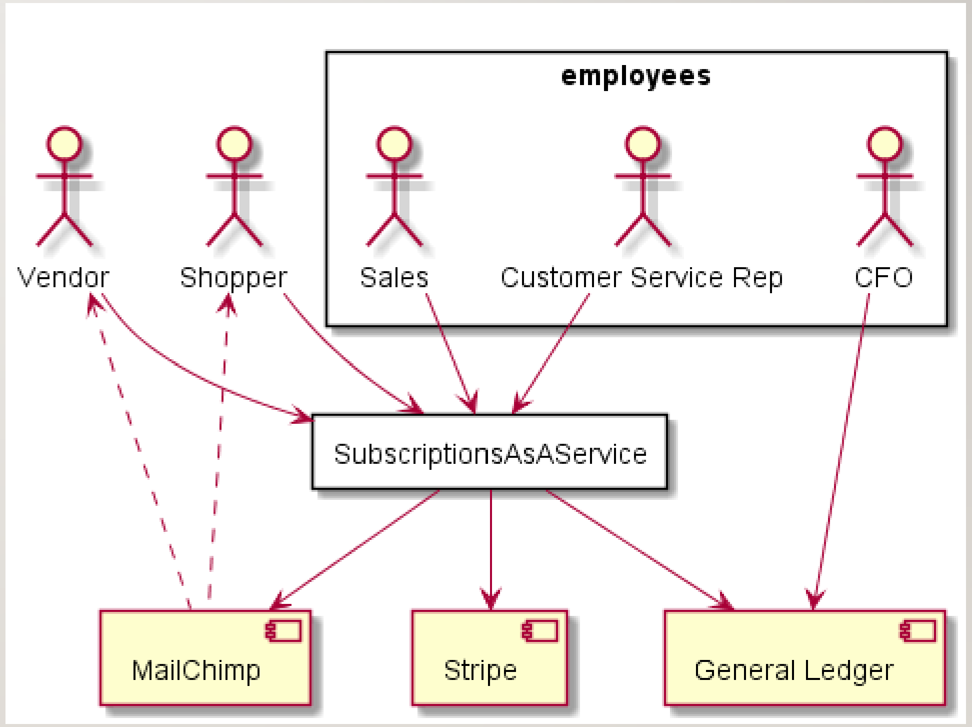
https://gitlab.com/mtnygard/arch-folder-template

DIAGRAMS FROM TEXT

- Graphics diff badly
- Specialized diagramming (Visio, Omnigraffle, SparxSystems) exclude people who don’t have licenses or don’t know how to use it.
- Try plain-text sources and text-to-image tools

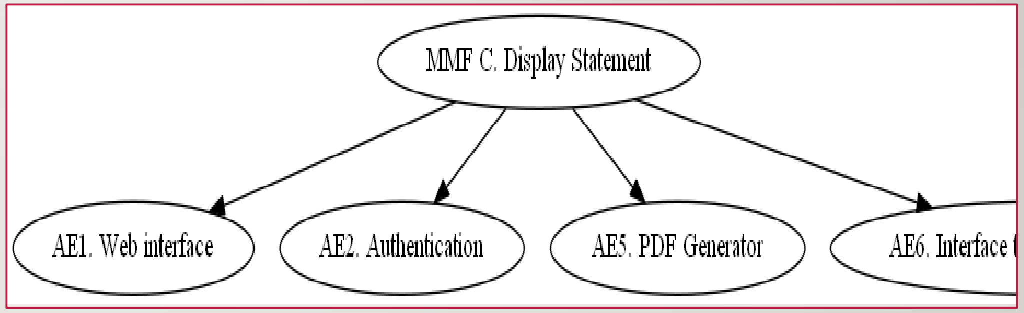
PLANTUML

```
"Shopper" as shopper
"Vendor" as vendor

rectangle "employees" {
  "Customer Service Rep" as csr
  "Sales" as sales
  "CFO" as cfo
}

[MailChimp] as mailer
[Stripe] as payments
[General Ledger] as gl

rectangle "SubscriptionsAsAService" as saas
{
```



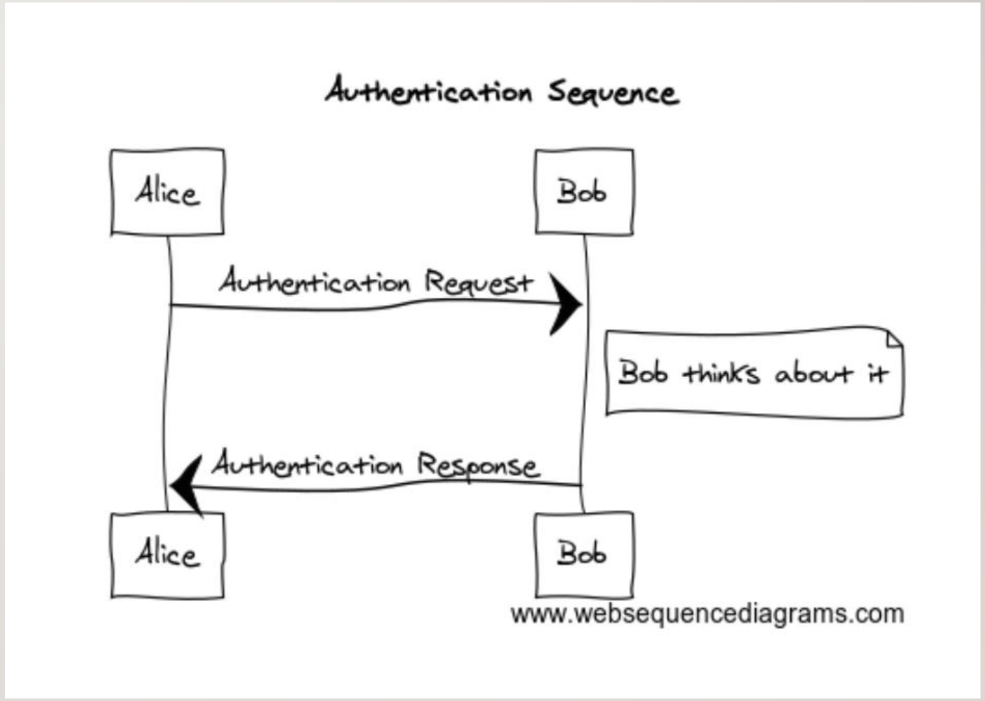
GRAPHVIZ

```
AE7 [label="AE7. Transaction Monitor"];
AE8 [label="AE8. Merchant Directory"];
MMF A [label="MMF A. Display Balance"];
MMF B [label="MMF B. Display Last 10
Transactions"];
MMF C [label="MMF C. Display Statement"];
MMF D [label="MMF D. Pay Bills"];
MMF C
MMF C  ->  AE1
MMF C  ->  AE2
MMF C  ->  AE5
MMF C  ->  AE6
MMF B
MMF B  ->  AE4
MMF A
MMF A  ->  AE3
```



WEB SEQUENCE DIAGRAMS

```
title Authentication Sequence

Alice->Bob: Authentication Request
note right of Bob: Bob thinks about it
Bob->Alice: Authentication Response
```



https://www.websequencediagrams.com/

COMMON THEME

- Spend little to no time pushing polygons
- Share thoughts
- Communicate to people
- Favor communication over completeness

SENDING MESSAGES TO THE FUTURE

THREE STAGES OF MEMORY LOSS

- “When can we remove this horrible hack?”

THREE STAGES OF MEMORY LOSS

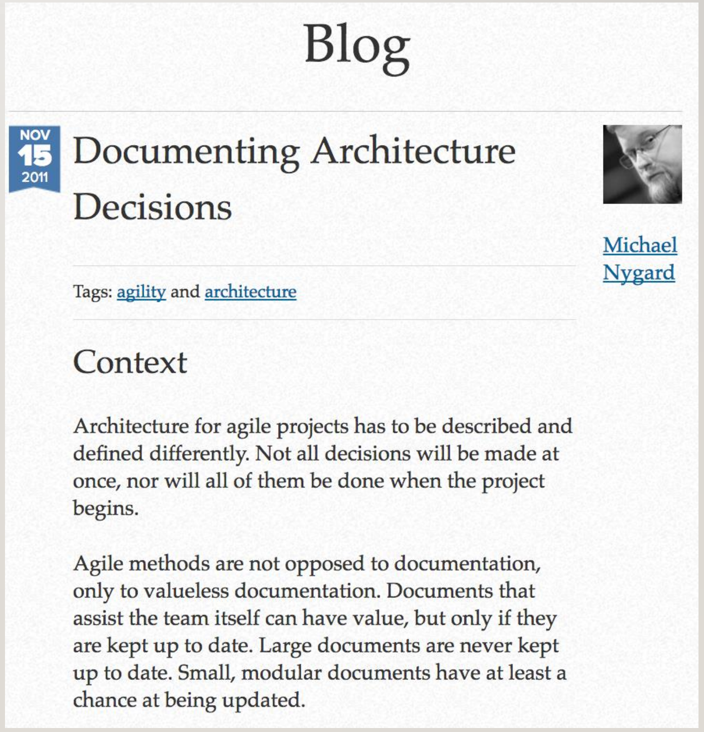
- “When can we remove this horrible hack?”
- “Does anyone know why we do this?” “Go ask Alice.”

THREE STAGES OF MEMORY LOSS

- “When can we remove this horrible hack?”
- “Does anyone know why we do this?” “Go ask Alice.”
- “Nobody knows why we do that, so I’m afraid to remove it.”

ARCHITECTURE DECISION RECORDS

- Point in time
- Rationale for a decision
- Immutable record
- But may be superseded later


Blog

NOV 15 2011

Documenting Architecture Decisions

Michael Nygard

Tags: agility and architecture

Context

Architecture for agile projects has to be described and defined differently. Not all decisions will be made at once, nor will all of them be done when the project begins.

Agile methods are not opposed to documentation, only to valueless documentation. Documents that assist the team itself can have value, but only if they are kept up to date. Large documents are never kept up to date. Small, modular documents have at least a chance at being updated.

KEYS TO A SUCCESSFUL ADR

- Think hard about the context
  - Include social, team, and skills factors
  - Include business situation, especially w.r.t. timelines
- Think hard about the consequences
  - Not “pros” and “cons”
  - Just this we need to do differently

40 lines (21 sloc) | 2.64 KB Raw Blame History

Architecture Decision Record: Simplification of Chimera Model

Note: this ADR supersedes some aspects of ADR-15 and ADR-16.

Context

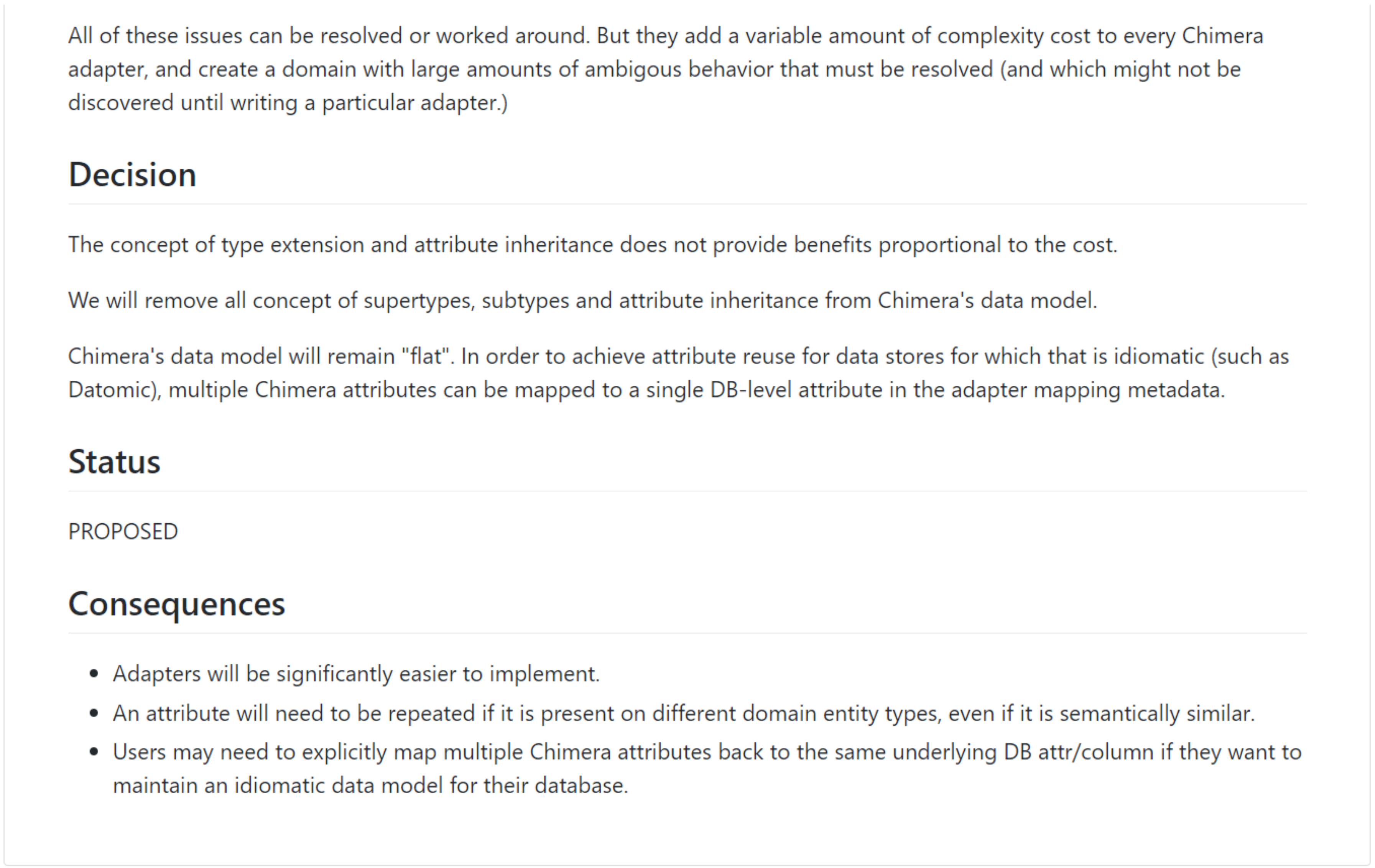
The Chimera data model (as described in ADR-15 and ADR-16) includes the concepts of *entity types* in the domain data model: a defined entity type may have supertypes, and inherits all the attributes of a given supertype

This is quite expressive, and is a good fit for certain types of data stores (such as Datomic, graph databases, and some object stores.) It makes it possible to compose types, and re-use attributes effectively.

However, it leads to a number of conceptual problems, as well as implementation complexities. These issues include but are not limited to:

- There is a desire for some types to be "abstract", in that they exist purely to be extended and are not intented to be reified in the target database (e.g, as a table.) In the current model it is ambiguous whether this is the case or not.
- A singe `extend-type` migration operation may need to create multiple columns in multiple tables, which some databases do not support transactionally.
- When doing a lookup by attribute that exists in multiple types, it is ambiguous which type is intended.
- In a SQL database, how to best model an extended type becomes ambiguous: copying the column leads to "denormalization", which might not be desired. On the other hand, creating a separate table for the shared columns leads to more complex queries with more joins.

All of these issues can be resolved or worked around. But they add a variable amount of complexity cost to every Chimera adapter, and create a domain with large amounts of ambigous behavior that must be resolved (and which might not be discovered until writing a particular adapter.)


All of these issues can be resolved or worked around. But they add a variable amount of complexity cost to every Chimera adapter, and create a domain with large amounts of ambigous behavior that must be resolved (and which might not be discovered until writing a particular adapter.)

Decision

The concept of type extension and attribute inheritance does not provide benefits proportional to the cost.

We will remove all concept of supertypes, subtypes and attribute inheritance from Chimera's data model.

Chimera's data model will remain "flat". In order to achieve attribute reuse for data stores for which that is idiomatic (such as Datomic), multiple Chimera attributes can be mapped to a single DB-level attribute in the adapter mapping metadata.

Status

PROPOSED

Consequences

- Adapters will be significantly easier to implement.
- An attribute will need to be repeated if it is present on different domain entity types, even if it is semantically similar.
- Users may need to explicitly map multiple Chimera attributes back to the same underlying DB attr/column if they want to maintain an idiomatic data model for their database.

https://github.com/arachne-framework/architecture/blob/master/adr-017-simplify-chimera-model.md

SAMPLE ADRs

https://github.com/arachne-framework/architecture

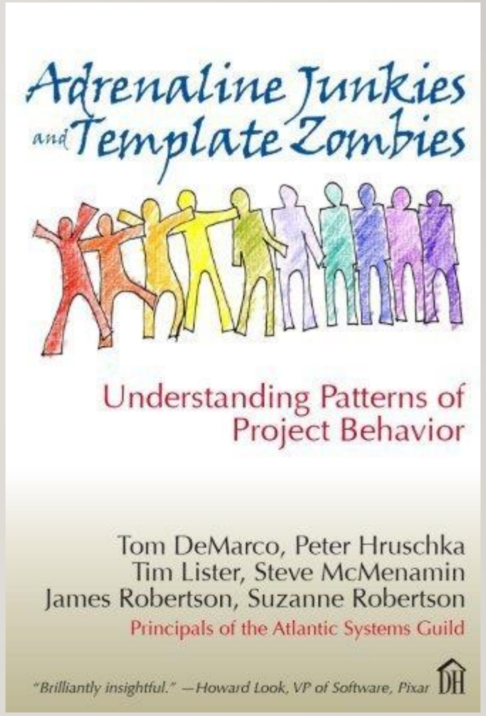
https://github.com/npryce/adr-tools

ACTIVITY: DISCUSSION

- Why is it important that an ADR isn’t trying to “sell” a decision?

REMINDERS

- We document to:
    1. Communicate ideas
    2. Capture information
    3. Aid our memory
    4. Share our thinking with the future
- Use the right views to serve different viewpoints
- Never be a template zombie.


Adrenaline Junkies and Template Zombies
Understanding Patterns of Project Behavior
Tom DeMarco, Peter Hruschka
Tim Lister, Steve McMenamin
James Robertson, Suzanne Robertson
Principals of the Atlantic Systems Guild
"Brilliantly insightful." —Howard Look, VP of Software, Pixar

NEXT UP: RESPONSIBILITY ALLOCATION