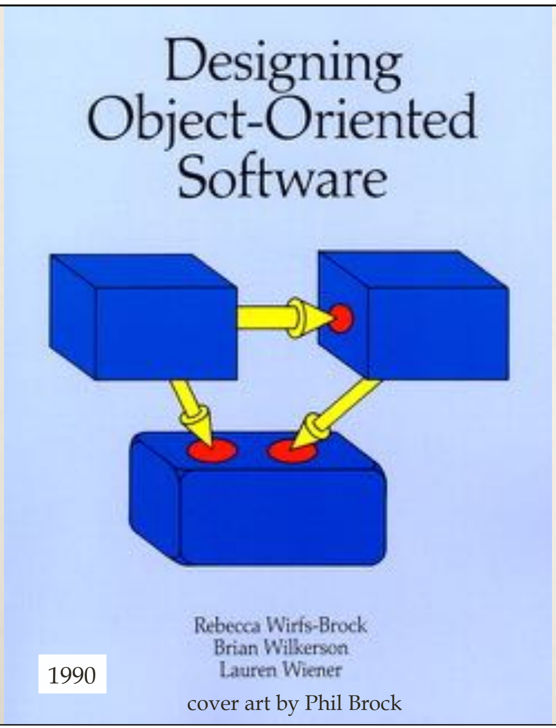
Cultivating Your Personal Design Heuristics

Rebecca Wirfs-Brock

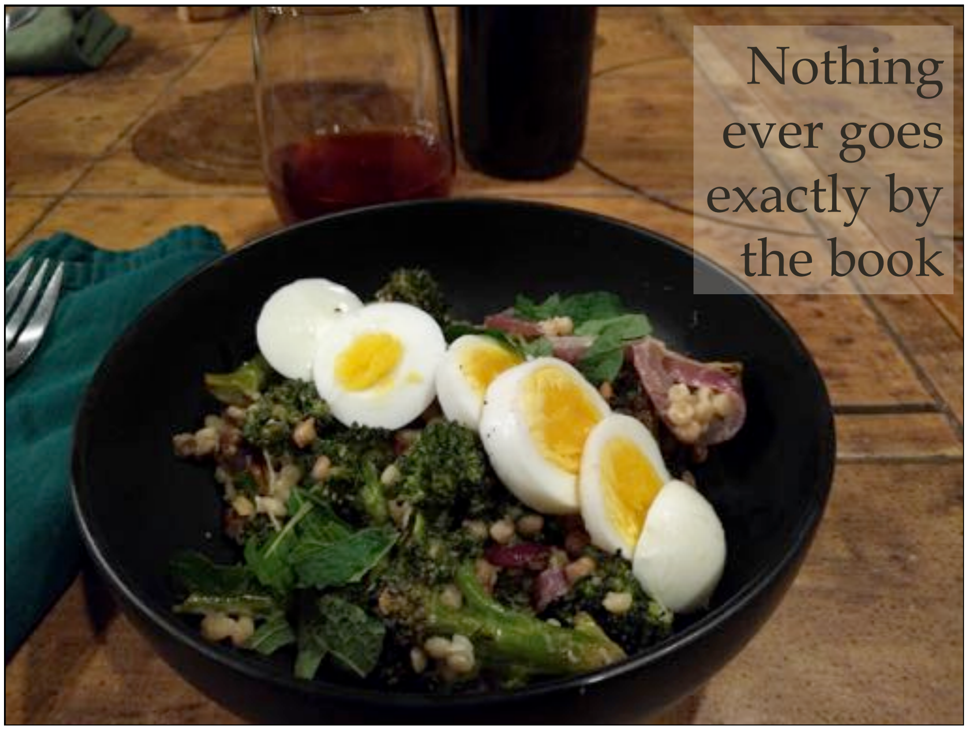


Designing Object-Oriented Software

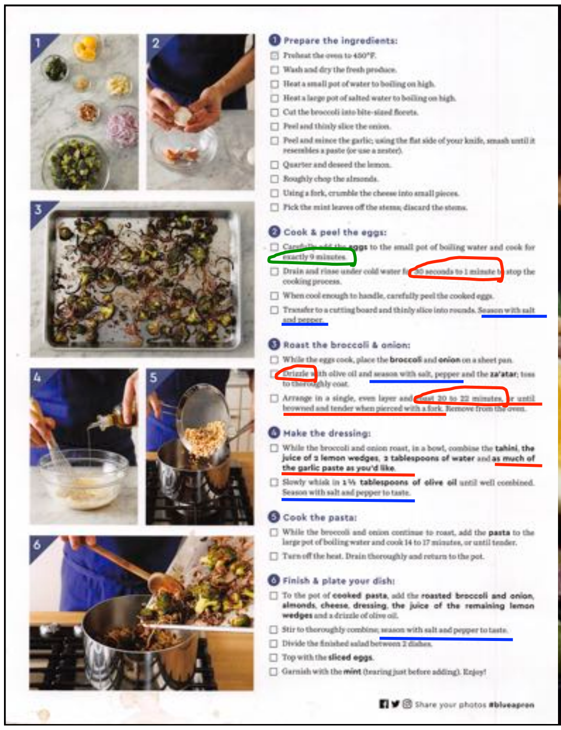
Rebecca Wirfs-Brock
Brian Wilkerson
Lauren Wiener

1990

cover art by Phil Brock



Nothing ever goes exactly by the book



1 Prepare the ingredients:

- ☐ Preheat the oven to 450°F.
- ☐ Wash and dry the fresh produce.
- ☐ Heat a small pot of water to boiling on high.
- ☐ Heat a large pot of salted water to boiling on high.
- ☐ Cut the broccoli into bite-sized florets.
- ☐ Peel and thinly slice the onion.
- ☐ Peel and mince the garlic; using the flat side of your knife, smash until it resembles a paste (or use a zester).
- ☐ Quarter and deseed the lemon.
- ☐ Roughly chop the almonds.
- ☐ Using a fork, crumble the cheese into small pieces.
- ☐ Pick the mint leaves off the stems; discard the stems.

2 Cook & peel the eggs:

- ☐ Carefully add the **eggs** to the small pot of boiling water and cook for exactly 9 minutes.
- ☐ Drain and rinse under cold water for 30 seconds to 1 minute to stop the cooking process.
- ☐ When cool enough to handle, carefully peel the cooked eggs.
- ☐ Transfer to a cutting board and thinly slice into rounds. Season with salt and pepper.

3 Roast the broccoli & onion:

- ☐ While the eggs cook, place the **broccoli** and **onion** on a sheet pan.
- ☐ Drizzle with olive oil and season with salt, pepper and the **za'atar**; toss to thoroughly coat.
- ☐ Arrange in a single, even layer and roast 20 to 22 minutes, or until browned and tender when pierced with a fork. Remove from the oven.

4 Make the dressing:

- ☐ While the broccoli and onion roast, in a bowl, combine the **tahini**, **the juice of 2 lemon wedges**, **2 tablespoons of water** and **as much of the garlic paste as you'd like**.
- ☐ Slowly whisk in **1½ tablespoons of olive oil** until well combined. Season with salt and pepper to taste.

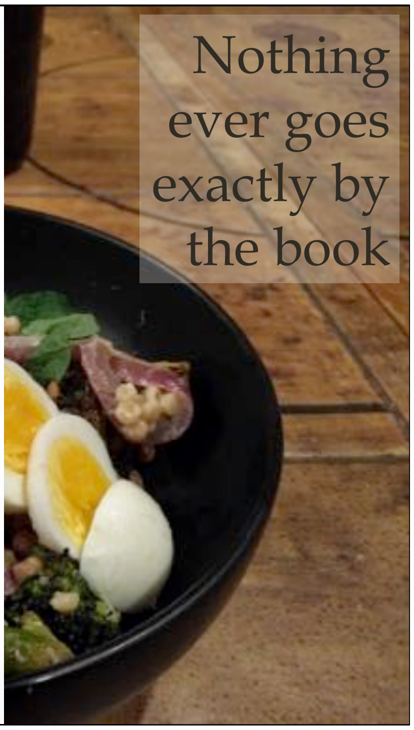
5 Cook the pasta:

- ☐ While the broccoli and onion continue to roast, add the **pasta** to the large pot of boiling water and cook 14 to 17 minutes, or until tender.
- ☐ Turn off the heat. Drain thoroughly and return to the pot.

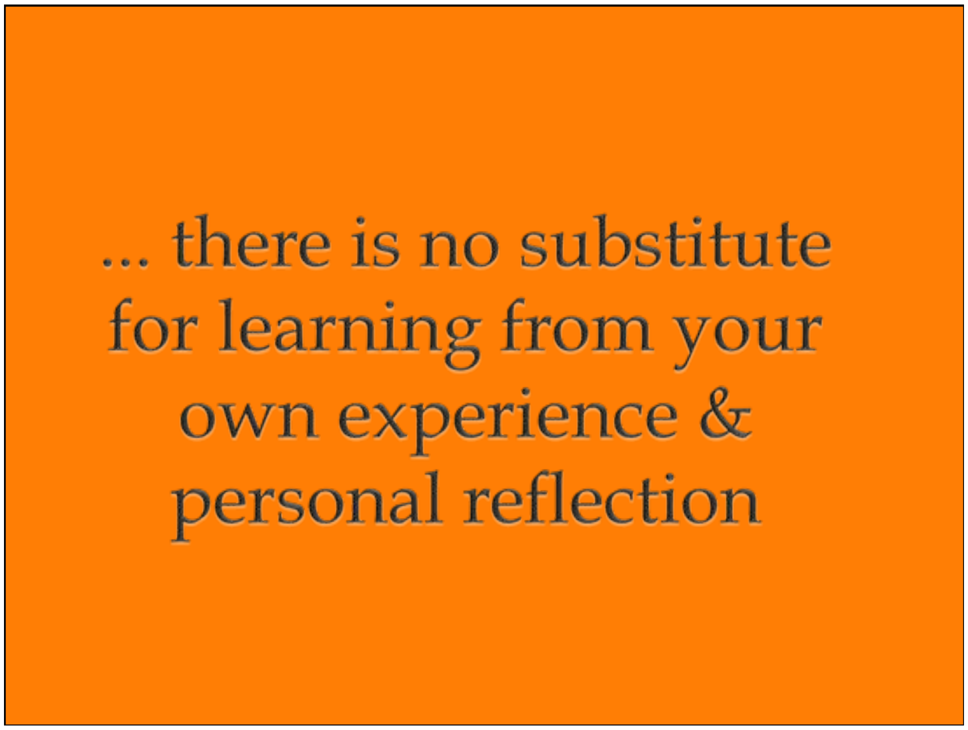
6 Finish & plate your dish:

- ☐ To the pot of **cooked pasta**, add the **roasted broccoli and onion**, **almonds**, **cheese**, **dressing**, **the juice of the remaining lemon wedges** and a drizzle of olive oil.
- ☐ Stir to thoroughly combine; season with salt and pepper to taste.
- ☐ Divide the finished salad between 2 dishes.
- ☐ Top with the **sliced eggs**.
- ☐ Garnish with the **mint** (tearing just before adding). Enjoy!

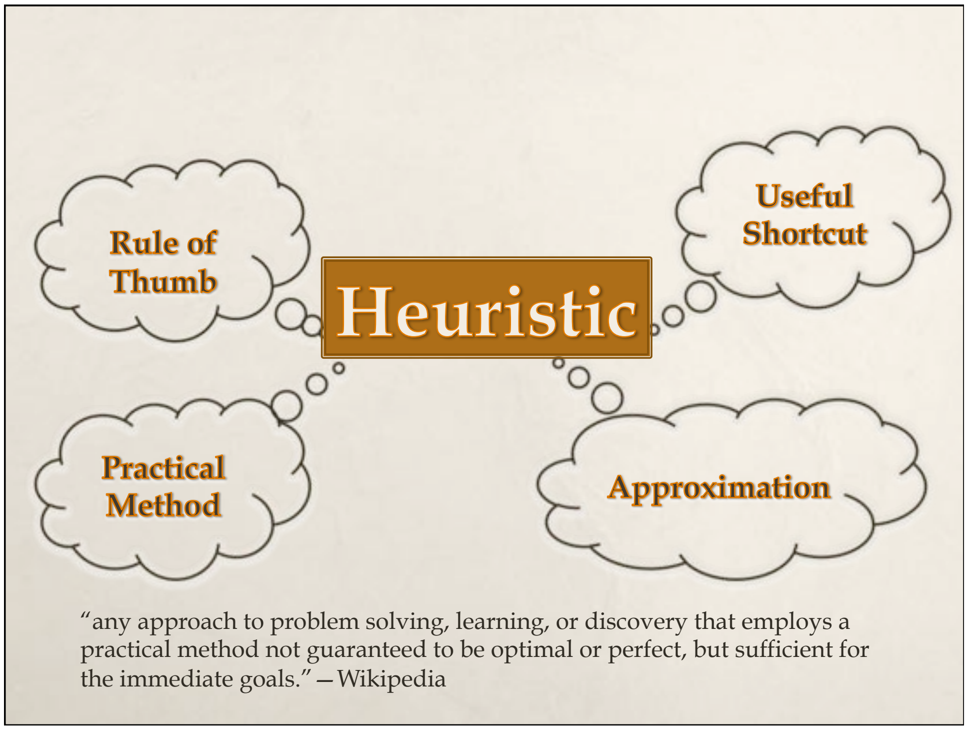
Share your photos **#blueapron**



Nothing ever goes exactly by the book



… there is no substitute for learning from your own experience & personal reflection

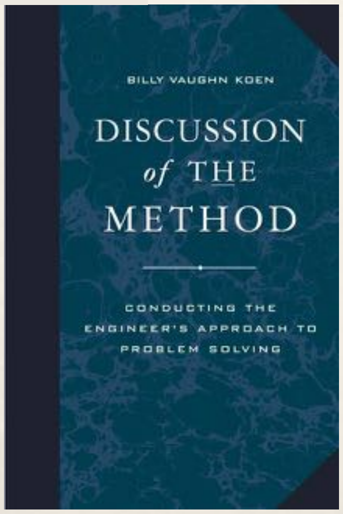


"any approach to problem solving, learning, or discovery that employs a practical method not guaranteed to be optimal or perfect, but sufficient for the immediate goals." —Wikipedia

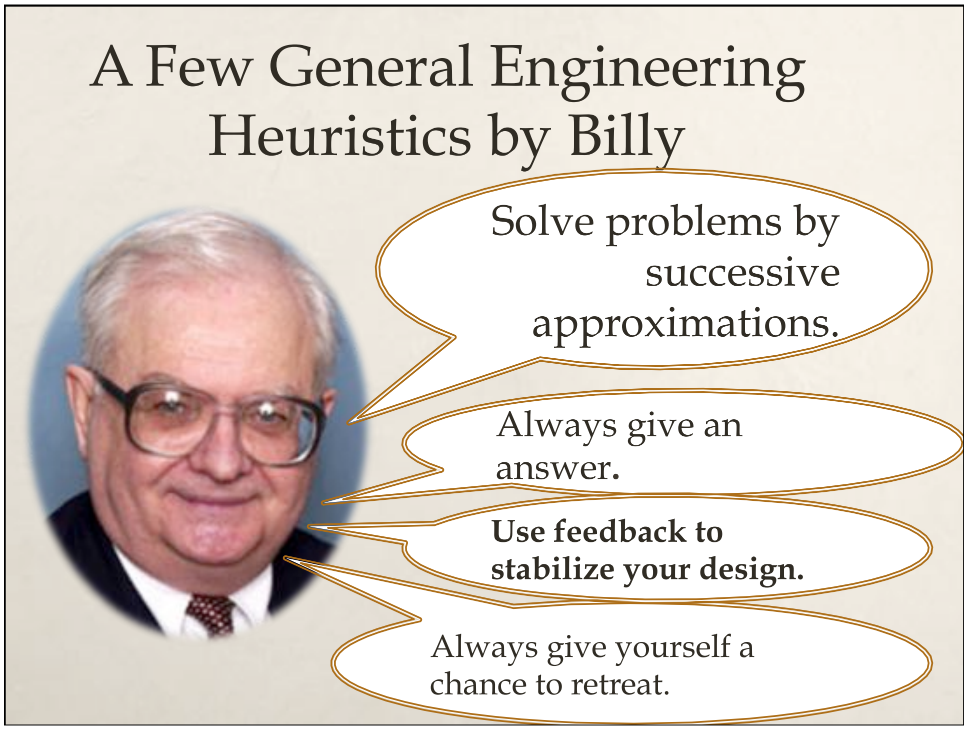
Heuristic

"anything that provides a plausible aid or direction in the solution of a problem but is in the final analysis unjustified, incapable of justification, and potentially fallible."

—Billy Vaughn Koen



What do typical heuristics look like?



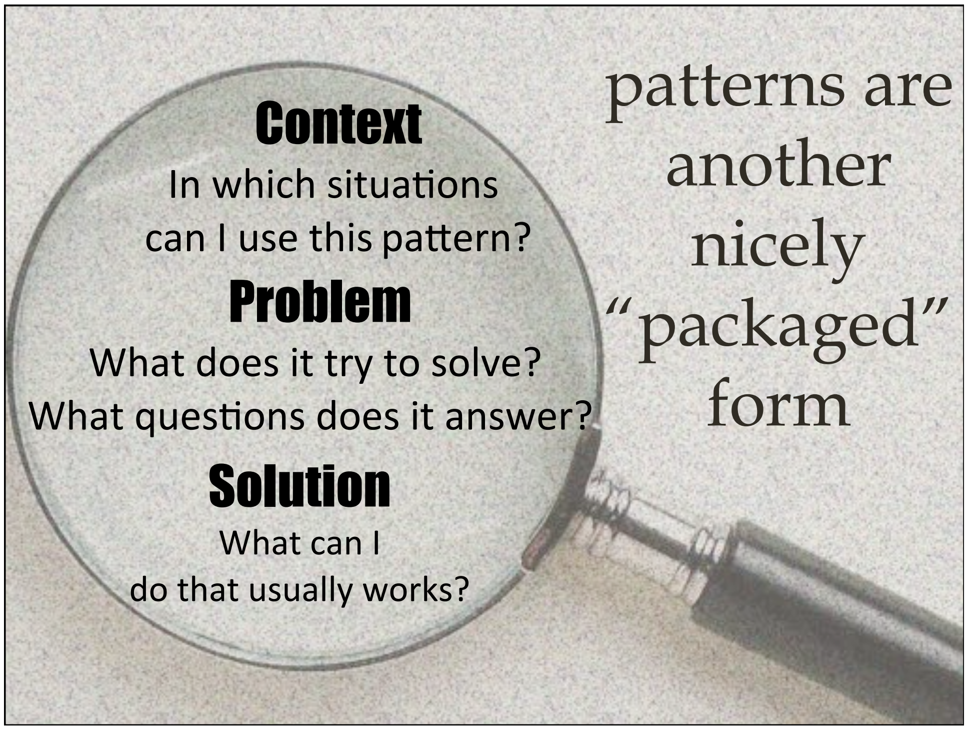
A Few General Engineering Heuristics by Billy

Solve problems by successive approximations.

Always give an answer.

Use feedback to stabilize your design.

Always give yourself a chance to retreat.



Context

In which situations can I use this pattern?

Problem

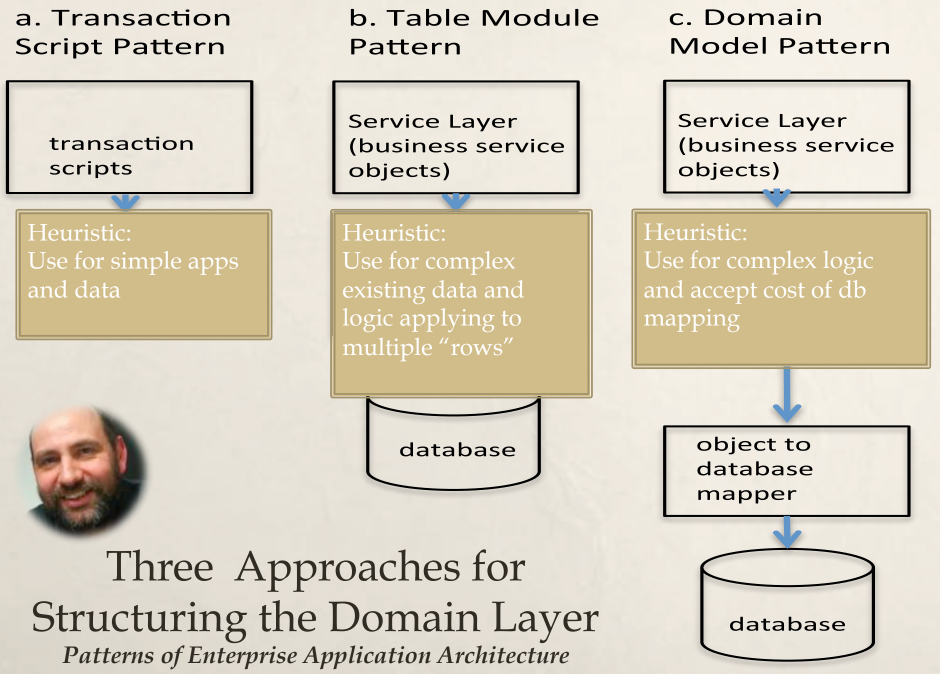
What does it try to solve?
What questions does it answer?

Solution

What can I do that usually works?

patterns are another nicely "packaged" form

1. Heuristics to Solve a Design Problem

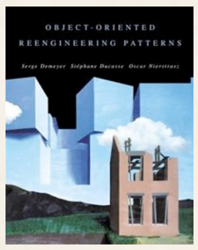

a. Transaction Script Pattern
transaction scripts
Heuristic: Use for simple apps and data
b. Table Module Pattern
Service Layer (business service objects)
Heuristic: Use for complex existing data and logic applying to multiple "rows"
database
c. Domain Model Pattern
Service Layer (business service objects)
Heuristic: Use for complex logic and accept cost of db mapping
object to database mapper
database

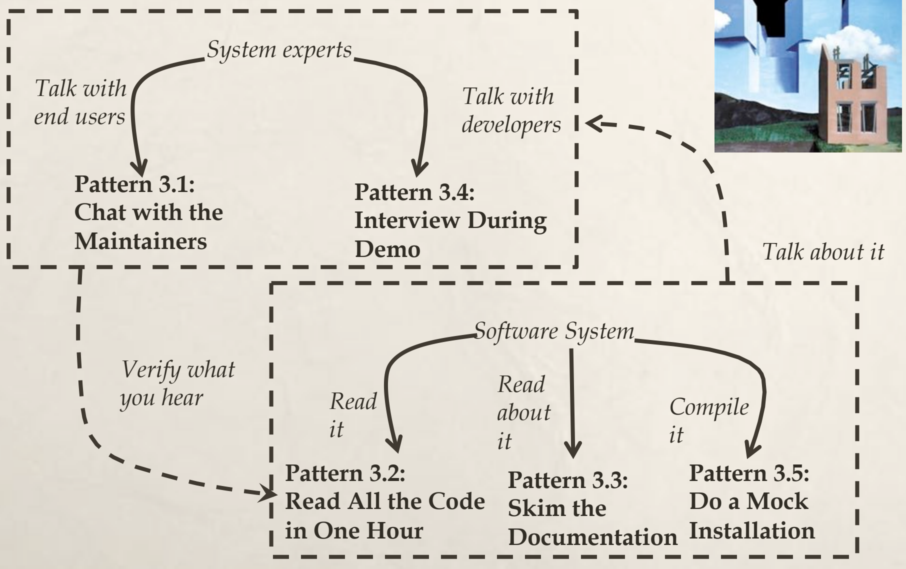
Three Approaches for Structuring the Domain Layer

Patterns of Enterprise Application Architecture

2.
Heuristics to Guide Use of Other Heuristics

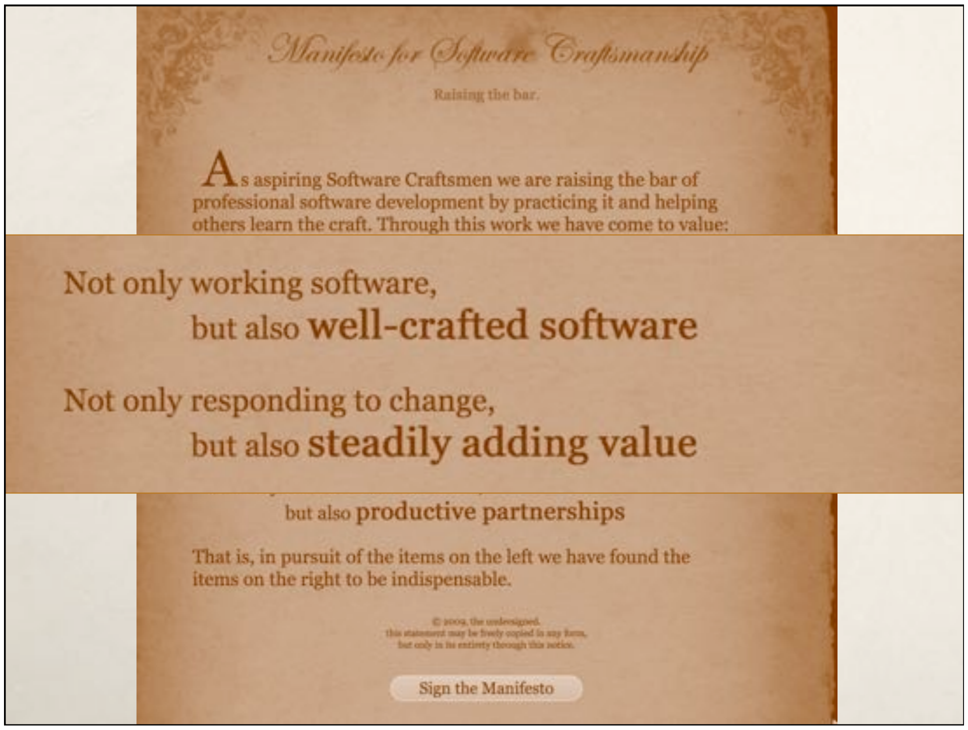
First Contact Patterns




System experts
Talk with end users
Talk with developers
Pattern 3.1: Chat with the Maintainers
Pattern 3.4: Interview During Demo
Talk about it
Verify what you hear
Software System
Read it
Read about it
Compile it
Pattern 3.2: Read All the Code in One Hour
Pattern 3.3: Skim the Documentation
Pattern 3.5: Do a Mock Installation

Each chapter in *Object-Oriented Reengineering Patterns* is a small language

3. Heuristics that Determine our Attitude and Behavior



Manifesto for Software Craftsmanship

Raising the bar.

As aspiring Software Craftsmen we are raising the bar of professional software development by practicing it and helping others learn the craft. Through this work we have come to value:

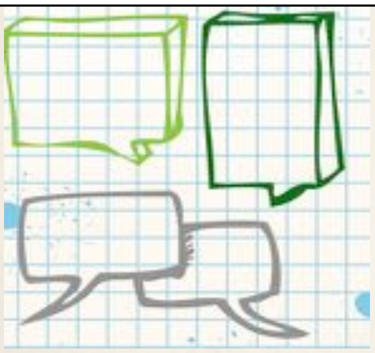
Not only working software,
but also **well-crafted software**

Not only responding to change,
but also **steadily adding value**

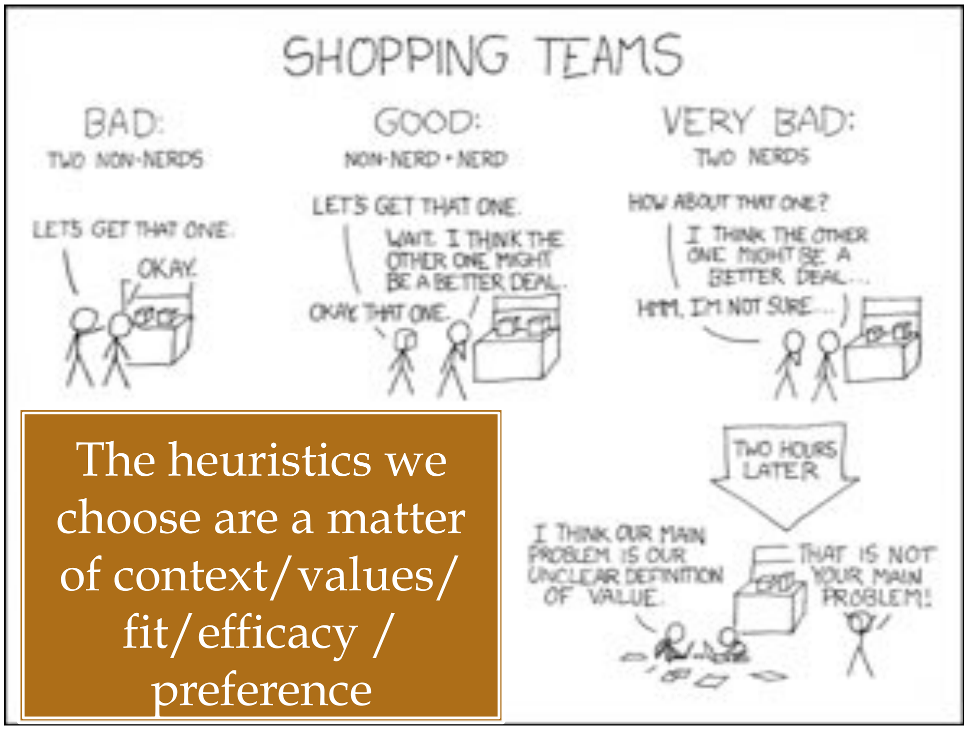
but also **productive partnerships**

That is, in pursuit of the items on the left we have found the items on the right to be indispensable.

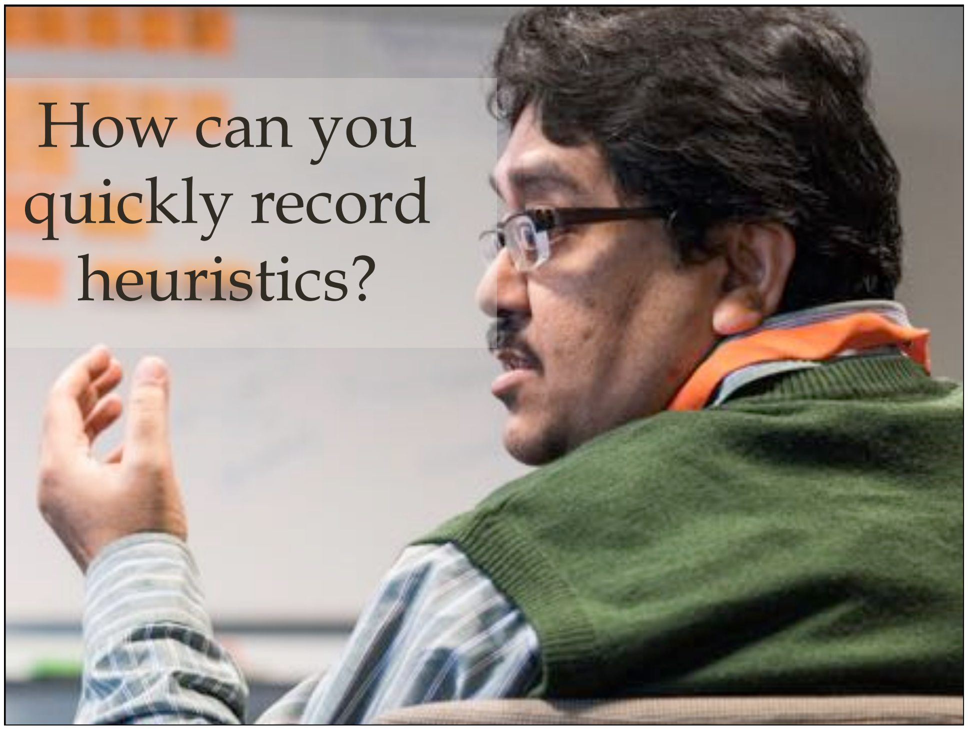
7 minute
Discussion



Share some "go to" design heuristics with your neighbor. Jot them down on stickies.


SHOPPING TEAMS
BAD:
TWO NON-NERDS
LET'S GET THAT ONE.
OKAY.
GOOD:
NON-NERD + NERD
LET'S GET THAT ONE.
WAIT. I THINK THE OTHER ONE MIGHT BE A BETTER DEAL.
OKAY. THAT ONE.
VERY BAD:
TWO NERDS
HOW ABOUT THAT ONE?
I THINK THE OTHER ONE MIGHT BE A BETTER DEAL...
HMM. I'M NOT SURE...
TWO HOURS LATER
I THINK OUR MAIN PROBLEM IS OUR UNCLEAR DEFINITION OF VALUE.
THAT IS NOT YOUR MAIN PROBLEM!

The heuristics we choose are a matter of context/values/ fit/efficacy / preference



How can you quickly record heuristics?

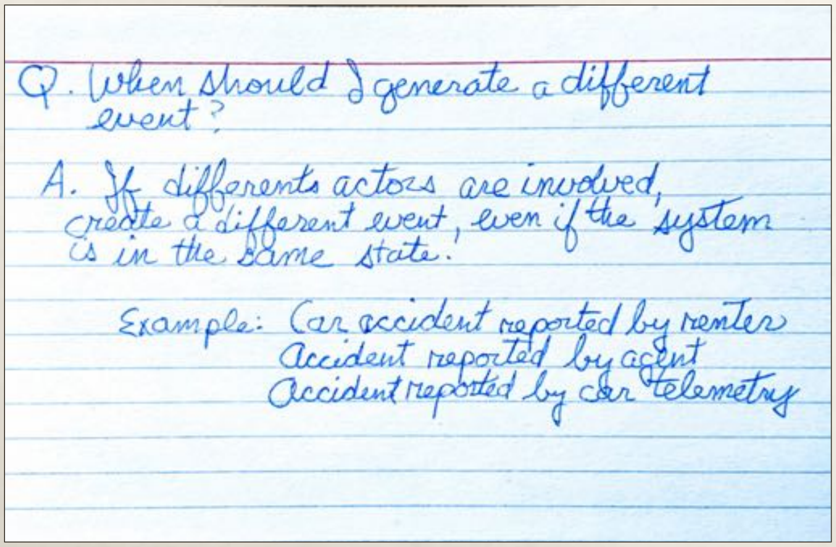
Avoid Modeling Gotchas

Identify Trusted Events

Happy Path + Expected Exception

Lazy microservices

Say more on
Question, Heuristic, Example (QHE) Cards



Q. When should I generate a different event?

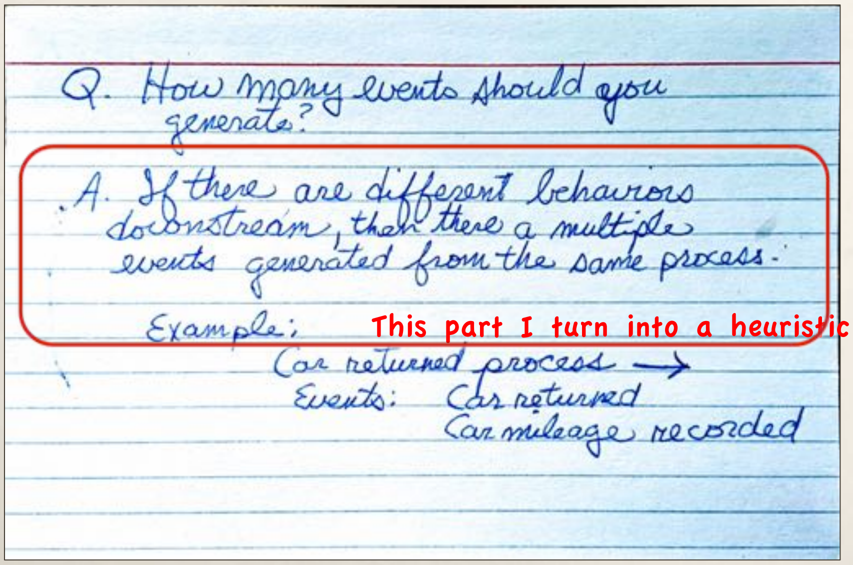
A. If differents actors are involved, create a different event, even if the system is in the same state.

Example: Car accident reported by renter
Accident reported by agent
Accident reported by car telemetry

Write a QHE Card. Ask the question, state the heuristic, then give at least 3 examples.

10 minute exercise

Question-Heuristic-Example Cards



Q. How many events should you generate?

A. If there are different behaviors downstream, then there a multiple events generated from the same process.

Example:

Car returned process →

Events: Car returned

Car mileage recorded

This part I turn into a heuristic

Heuristic: Generate different events for a business process if different downstream business processes react differently.

Working together, turn your QHE questions and answers into Heuristic statements

Pair or small team

Our State of The Art *(SOTA)*

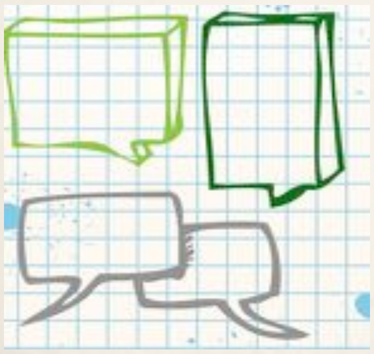
According to Vaughn Koen

- We each have our own cherished heuristics
- As new ones become useful we add to our collection
- No longer useful ones fall out of fashion
- **Make small changes to your state-of-the-art**
- **Sometimes, even useful ones fade away**

HOTTEST EDITORS
1995
2000
EMACS-VIM EDITOR WAR
2005 VIM
2010 NOTEPAD++
2015 SUBLIME TEXT
2020 CRISPR
2025 CRISPR (VIM KEYBINDINGS)

https://xkcd.com/1823/

Some Age Well!

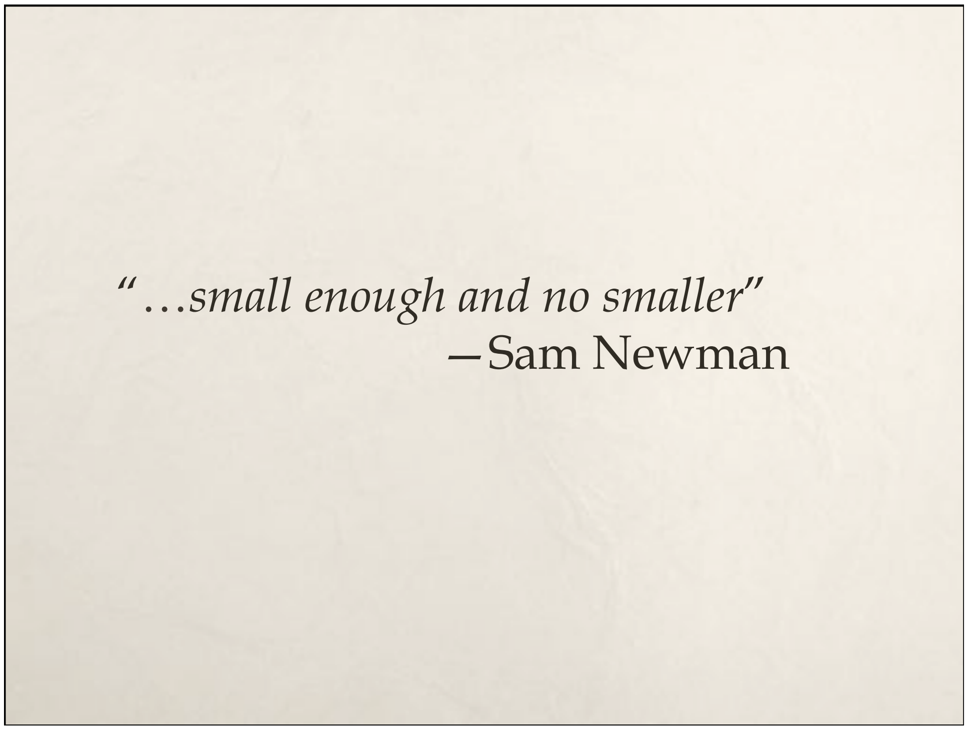


How have your heuristics have evolved?

Short discussion

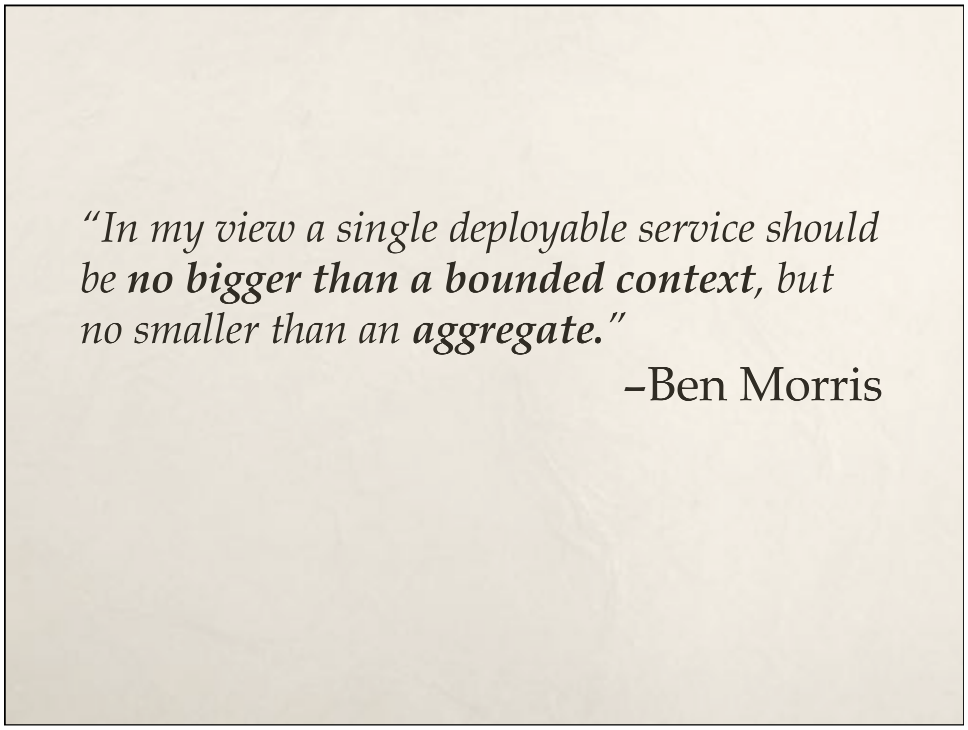
Heuristics Need to be Challenged

How Big Should a Microservice Be?



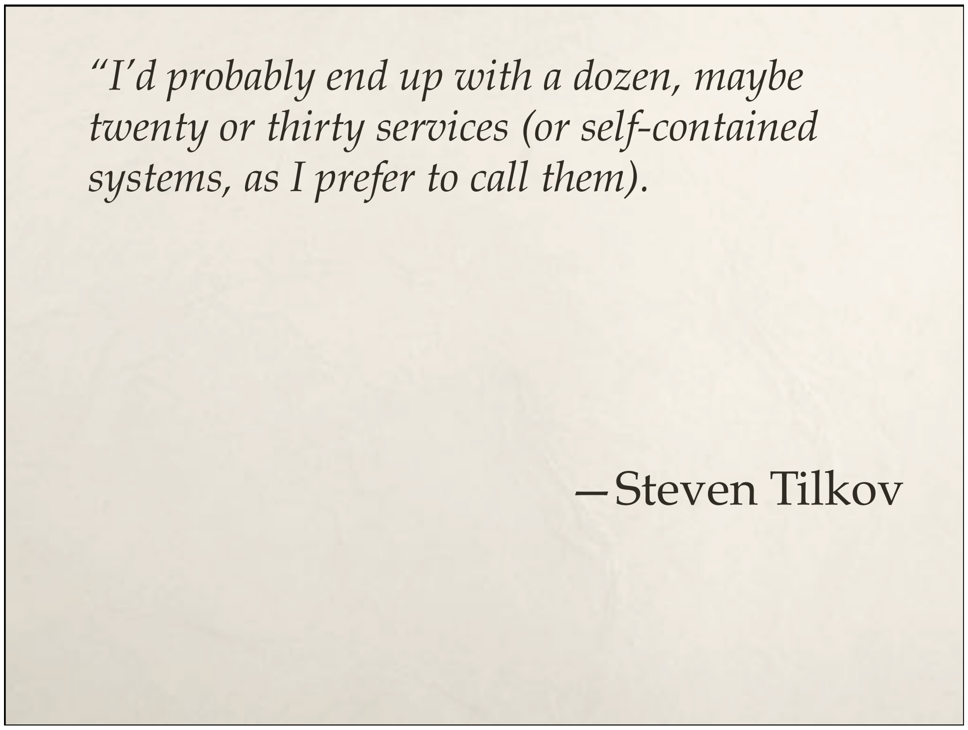
"…small enough and no smaller"

—Sam Newman



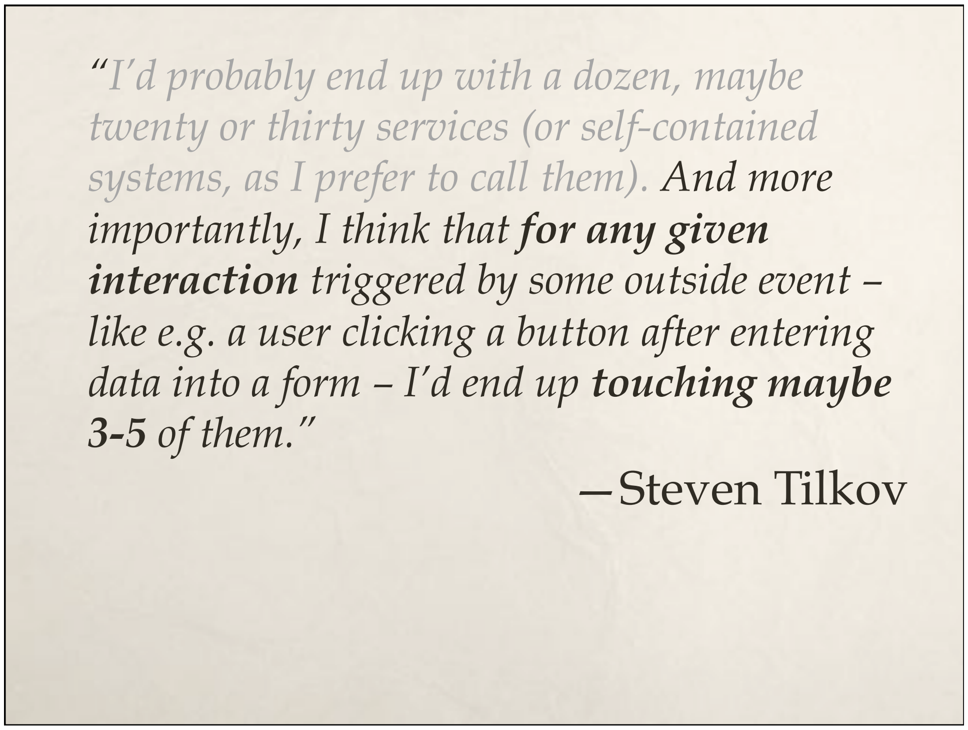
*"In my view a single deployable service should be **no bigger than a bounded context**, but no smaller than an **aggregate.**"*

–Ben Morris



"I'd probably end up with a dozen, maybe twenty or thirty services (or self-contained systems, as I prefer to call them).

—Steven Tilkov



*"I'd probably end up with a dozen, maybe twenty or thirty services (or self-contained systems, as I prefer to call them). And more importantly, I think that **for any given interaction** triggered by some outside event – like e.g. a user clicking a button after entering data into a form – I'd end up **touching maybe 3-5** of them."*

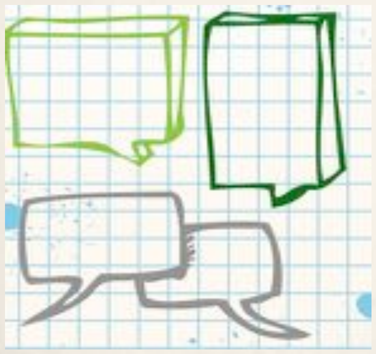
—Steven Tilkov

"Single Responsibility Principle: there should only be a single service impacted by a change to the definition of this data.
As a result, you'll tend to see services that aren't all that small, and probably not so many of them. In my experience, I've seen between 7 and 15 services the majority of the time."

—Udi Dahan

Heuristics Often Conflict…

Choose the heuristic to use from what ***you*** take to be the best option at the time you are required to choose.



Identify some competing heuristics

5 minute discussion

Techniques for Actively Cultivating Your Heuristics

Map out your interests

Microservice architectures

Tactical Design Heuristics for Functional Programming

Event Sourcing Architecture Heuristics

Model Understanding Heuristics

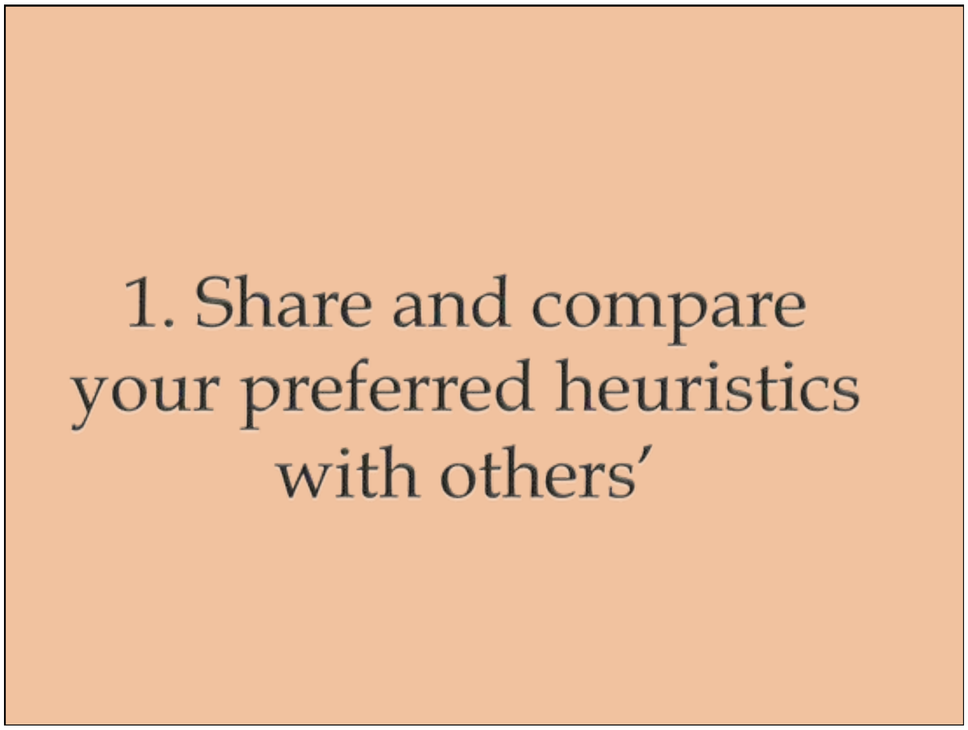
Design Nudging Heuristics

Where to next?

Microservice architectures

Model Understanding Heuristics

Design Nudging Heuristics



1. Share and compare your preferred heuristics with others'

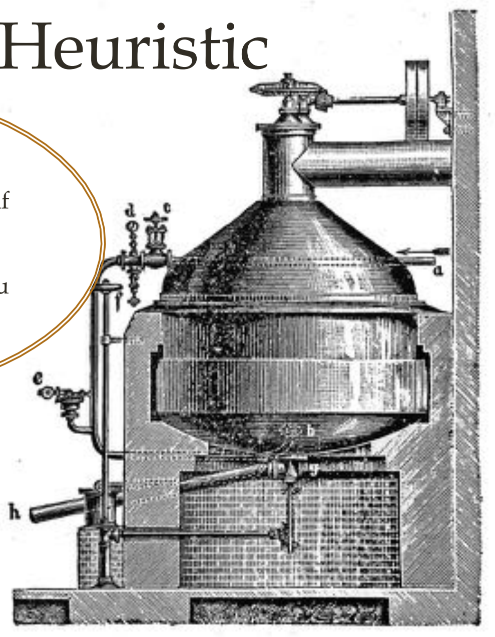
2. Take a view contrary to your preferred ways of working and argue both for and against it.

"As a rule, the more demanding the application, the more leverage you get from using a powerful language. But plenty of projects are not demanding at all. Most programming probably consists of writing little glue programs, and for little glue programs you can use any language that you're already familiar with and that has good libraries for whatever you need to do"

— Paul Graham, Revenge of the Nerds

Paul's Heuristic

It doesn't matter what programming language you use if you have a simple program. Use programming languages, tools, and frameworks and libraries you are familiar with.



© Can Stock Photo / gameover

My Imaginary Deb Paul

But Paul, what about the heuristic, use a rich domain model when you have rich behavior in your application?

And, use transaction scripts for really simple stuff that isn't going to change much.

And my lifelong heuristic:
Learn something new. Don't always do things the same way. That's soul sucking!

3. Have a conversation about a specific topic

My First Heuristics Distillation Conversation with Mathias Verraes

What's a heuristic you use when you model events?

Heuristic: Events are records of things that have happened, not things that will happen future.

The event is "a reservation has been made" or "service has been scheduled"

Examples Keep the Conversation Flowing

Here's another heuristic: A bounded context should keep its internal details private.

Say if you keep monetary units with 10 digits precision internally in a service, pass out an amount with 2 digits precision because that's all other consumers of the event would need.

We Dig Deeper…

Perhaps there's another heuristic?

Design agreed upon standard formats based on standard usage.

Don't design message or event contents for specific subscribers to that event?

And then it got really interesting…

What happens if a new process needs extra precision?

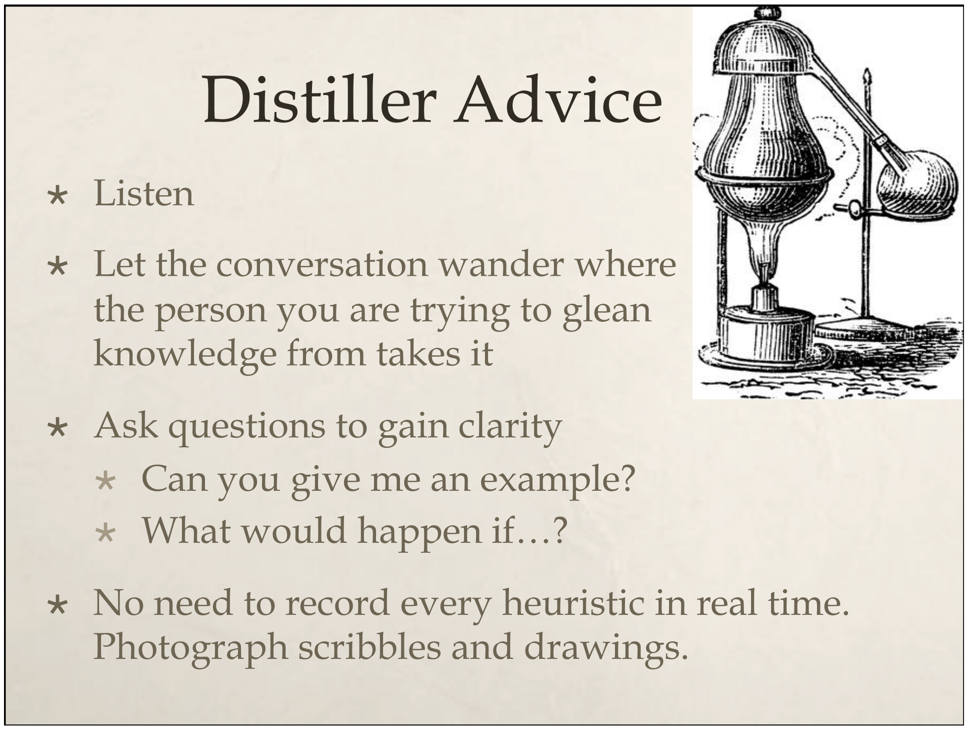
Maybe it belongs within the bound context of the process that knows 10 digits precision?

Which led us to this insight…

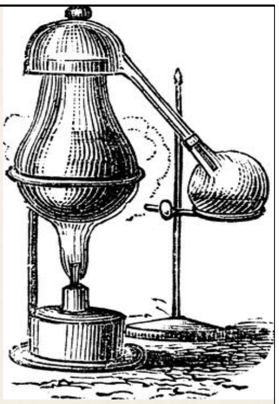
These two heuristics compete

Heuristic:
When designing information in an event, don't lose necessary precision.

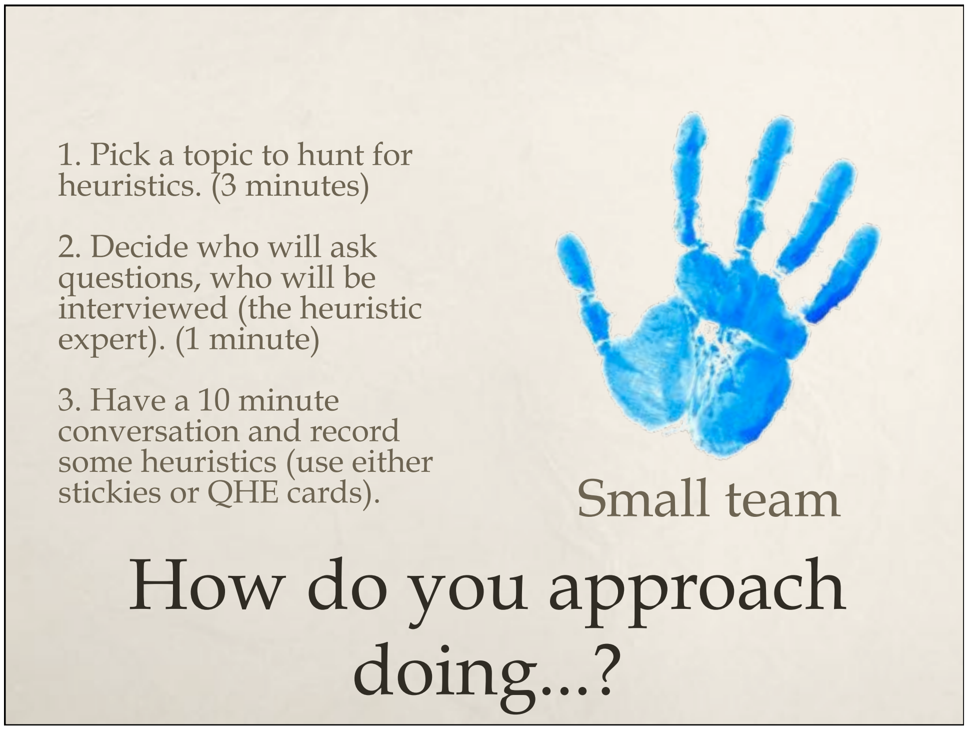
Heuristic:
Design agreed upon standard formats based on expected usage.



Distiller Advice



- Listen
- Let the conversation wander where the person you are trying to glean knowledge from takes it
- Ask questions to gain clarity
  - Can you give me an example?
  - What would happen if…?
- No need to record every heuristic in real time. Photograph scribbles and drawings.



1. Pick a topic to hunt for heuristics. (3 minutes)

2. Decide who will ask questions, who will be interviewed (the heuristic expert). (1 minute)

3. Have a 10 minute conversation and record some heuristics (use either stickies or QHE cards).

Small team

How do you approach doing...?

Heuristic: Generate different events for a business process if different downstream business processes react differently.

Heuristic Gists*

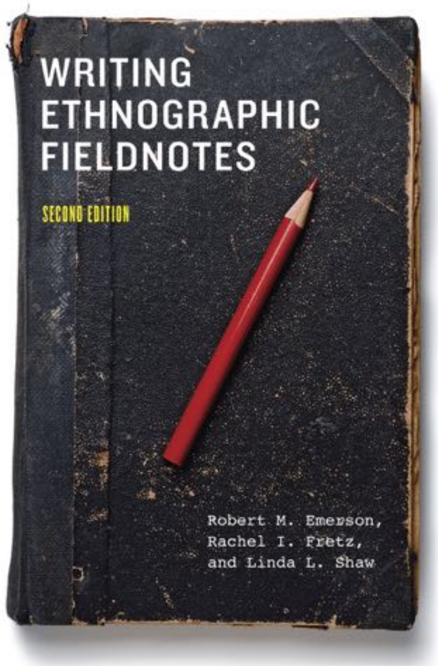
Multiple Events for a Single Process
You need to balance passing along information needed by downstream processes in a single business event with creating multiple event records, each designed to convey specific information needed by a specific downstream process.

Summary of Problem
How do you know how many events to generate from a single business process?

Summary of Solution
If different processes downstream react differently, generate different events. For example, handling a "rental car return" request might generate two events and event records: "car returned" and "mileage recorded." Even though the mileage is recorded at the time a car is returned, mileage could be recorded at any other time as well. It is a cleaner design to generate two events, rather than cram information into a single, overloaded "car returned" event.

*gist – the main point or part; essence. Similar to pattern thumbnails.

4. *Radical Idea:* Take notes of how you actually work


WRITING ETHNOGRAPHIC FIELDNOTES
SECOND EDITION
Robert M. Emerson, Rachel I. Fretz, and Linda L. Shaw

When?

as you attempt something new

you have a ½ hour

Distill what you do:

Record Your Design Values & Practices

61

Distill what you decide:

Document Design Decisions*

Title

Context - Forces at play

Decision - Stated with active voice: "We will ..."

Status - “proposed” or “accepted” later may be “deprecated” or “superseded”

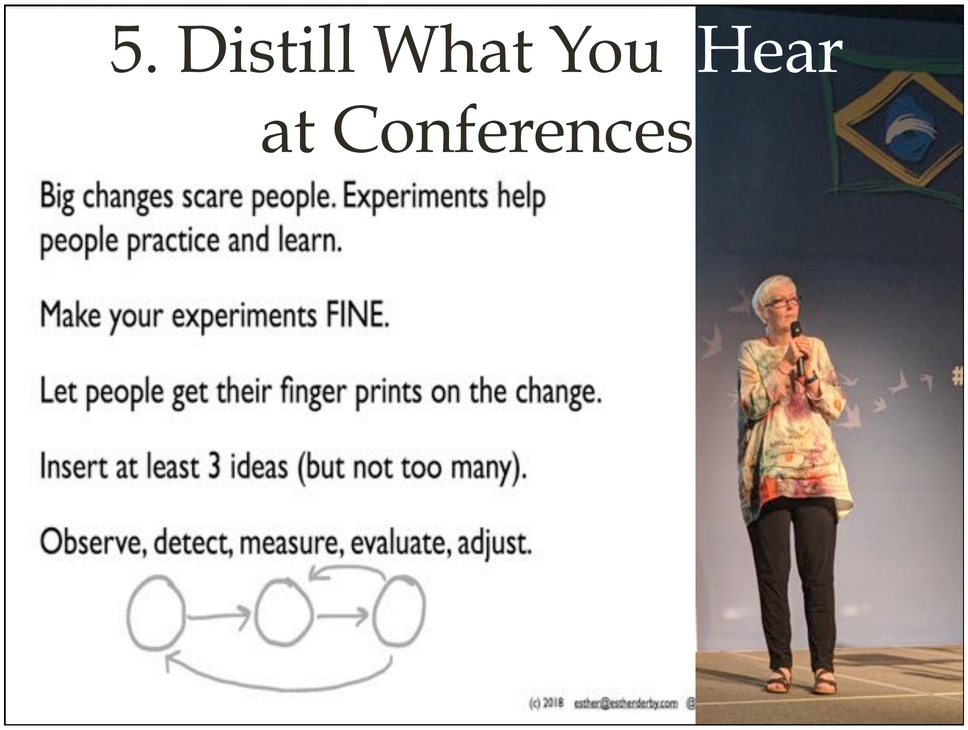
Consequences positive, negative, and neutral that affect the team and project in the future

*Thanks to Michael Nygard
http://thinkrelevance.com/blog/2011/11/15/documenting-architecture-decisions
Useful link to github project on decision records: https://github.com/joelparkerhenderson

5. Distill What You Hear at Conferences

Competing Event Source Evolution Heuristics
Michiel Overeem

5. Distill What You Hear at Conferences



Big changes scare people. Experiments help people practice and learn.

Make your experiments FINE.

Let people get their finger prints on the change.

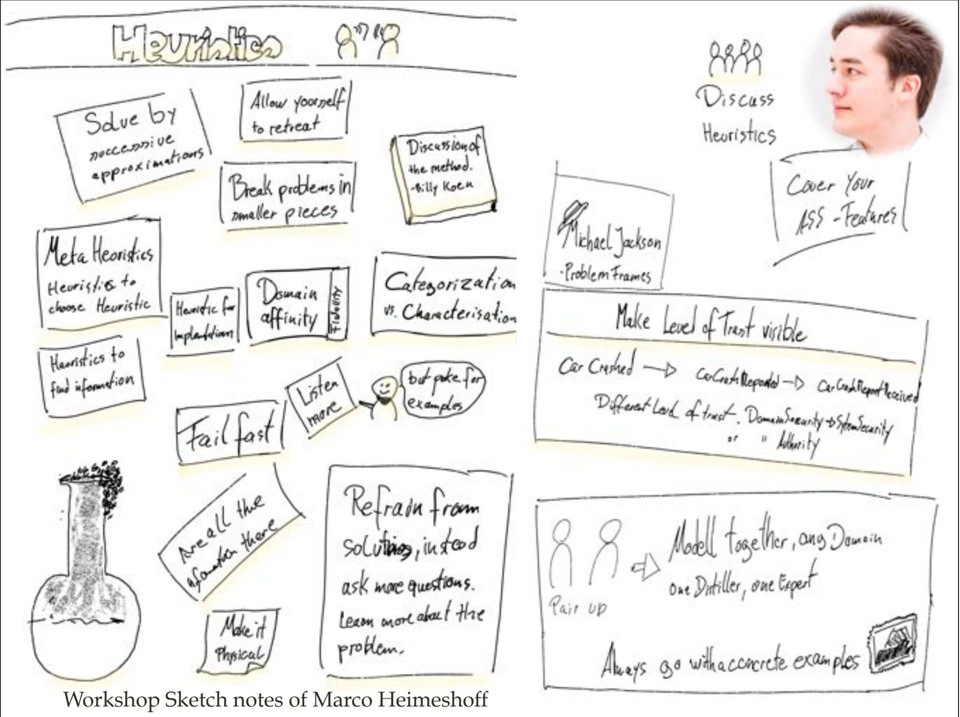
Insert at least 3 ideas (but not too many).

Observe, detect, measure, evaluate, adjust.

(c) 2018 esther@estherderby.com

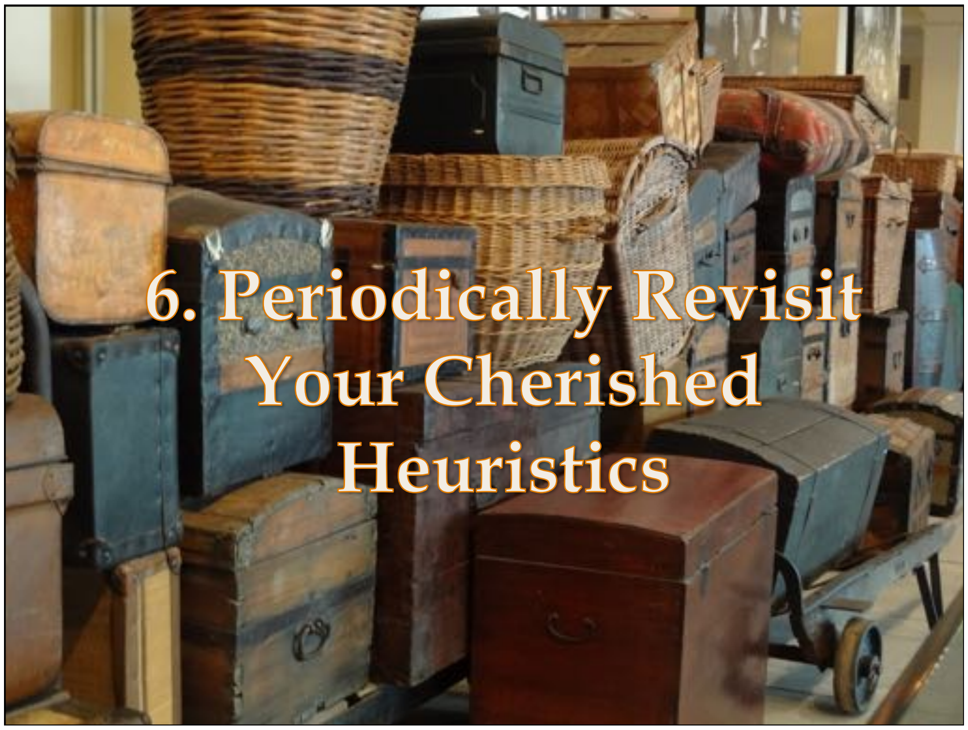
5. Distill What You Hear at Conferences

Produce tension with awkward examples.
Generate variation. Look for 'productive' models.
Introduce rigor.
Play in code.
Practice modeling!
Drill into one domain for a while.



Heuristics
Solve by successive approximations
Allow yourself to retreat
Break problems in smaller pieces
Discussion of the method. -Billy Koen
Meta Heuristics
Heuristics to choose Heuristic
Heuristic for implantation
Domain affinity
Fidelity
Categorization vs. Characterisation
Heuristics to find information
Fail fast
Listen more
but poke for examples
Are all the information there
Make it Physical
Refrain from solutions, instead ask more questions. Learn more about the problem.
Discuss Heuristics
Cover Your ASS - Features
Michael Jackson -Problem Frames
Make Level of Trust visible
Car Crashed → CarCrashReported → CarCrashReportReceived
Different Level of trust. DomainSecurity ⇒ SystemSecurity
or " Authority
Pair up
Modell together, any Domain
One Distiller, one Expert
Always go with a concrete examples

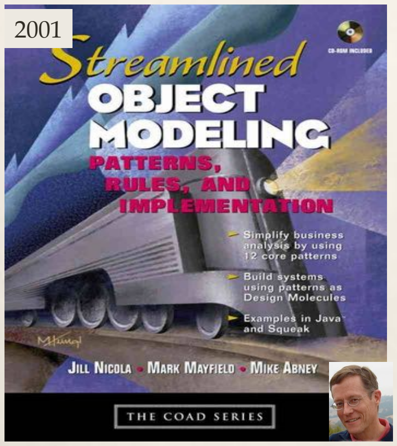
Workshop Sketch notes of Marco Heimeshoff



6. Periodically Revisit Your Cherished Heuristics

One of My Heuristics: By characterizing a domain entity's attributes you can identify needed system behaviors

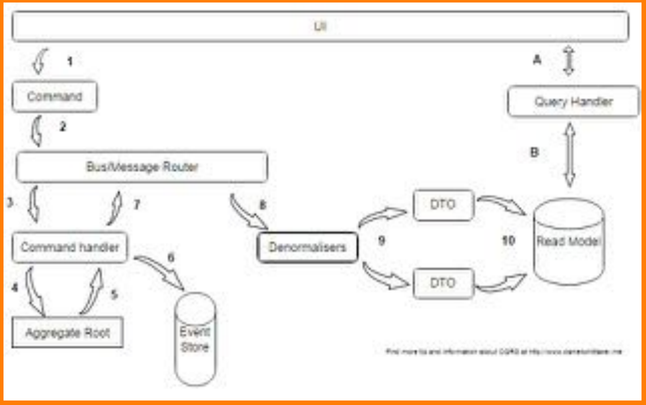
- *Descriptive Attributes* reflect a domain's properties (not identity).
- *Time-dependent attributes* Where maintaining a history of past values is important.
- *Lifecycle state attributes* Some entities go through a one-way lifecycle, from initial to final state.
- *Operational state* Some entities switch between different states. The state it is currently in determines how it behaves.


2001
Streamlined
OBJECT MODELING
PATTERNS, RULES, AND IMPLEMENTATION
Simplify business analysis by using 12 core patterns
Build systems using patterns as Design Molecules
Examples in Java and Squeak
JILL NICOLA · MARK MAYFIELD · MIKE ABNEY
THE COAD SERIES

Some of My Cherished Heuristics for Validating Data

- Perform simple edits (syntactic) in browser code
- Don't always trust browser-validated edits.
- Reapply them if receiving requests from an untrusted source
- Consistently assign validation responsibilities to framework-specific validation classes
- Consistently use domain validation and constraint enforcement patterns

…what's different about validating/enforcing constraints within a CQRS architecture?



Heuristic*:
Distinguish between "superficial" and "domain" validations and handle them differently

"superficial": what must be true, regardless of the state of the domain

Heuristic: Validate these before issuing a command, ideally on the client side as well as the server side

"superficial" but requires lookup of other information

Heuristic: Validate in the service before invoking the command

"domain": validity of a command is dependent on the state of the model

Heuristic: Validate in domain objects

*http://danielwhittaker.me/2016/04/20/how-to-validate-commands-in-a-cqrs-application/

Sorting out heuristics…

superficial vs. domain validations

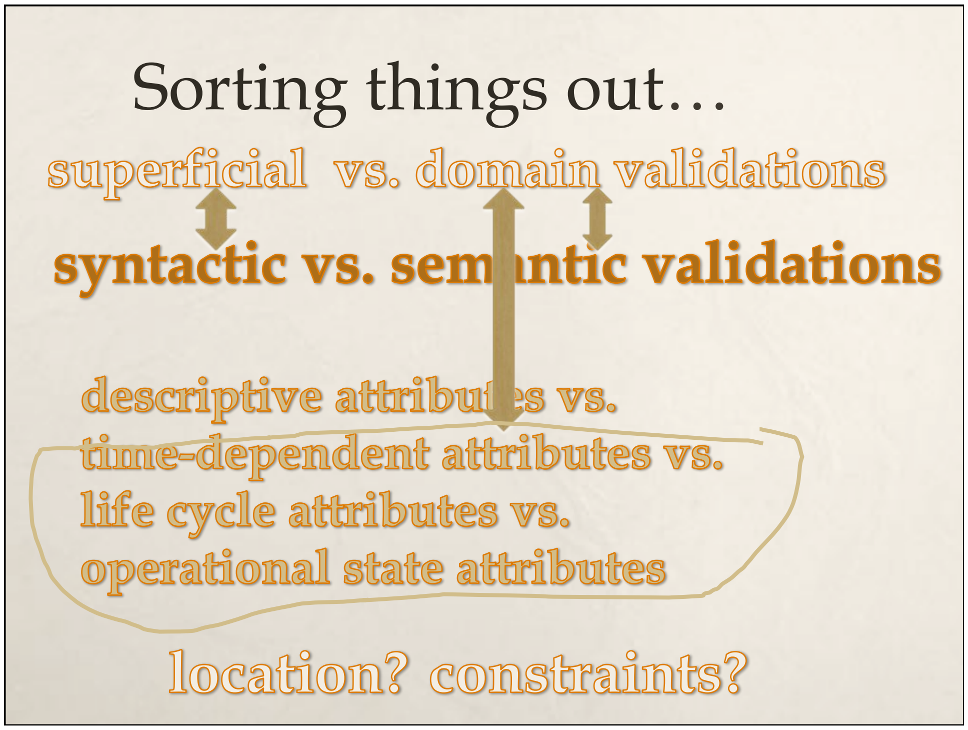
syntactic vs. semantic validations

descriptive attributes vs.
time-dependent attributes vs.
life cycle attributes vs.
operational state attributes

location? constraints?

Let's Just Get On With It

© Can Stock Photo / Zinkevych



Sorting things out…

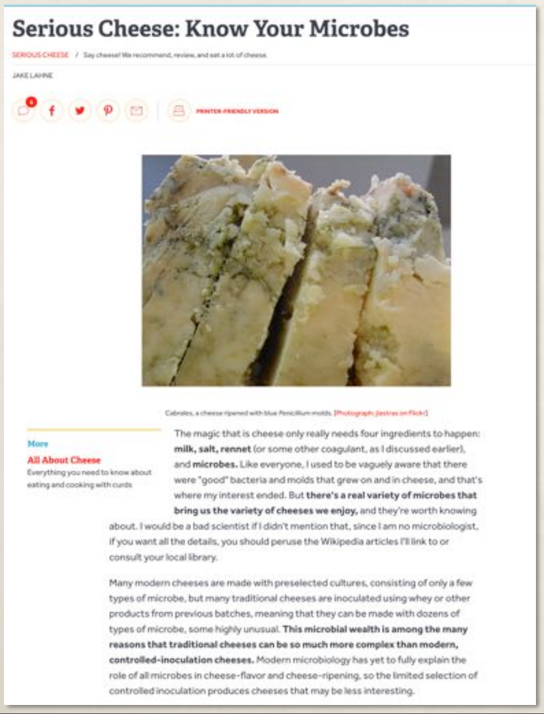
superficial vs. domain validations

$\updownarrow$

syntactic vs. semantic validations

descriptive attributes vs.
time-dependent attributes vs.
life cycle attributes vs.
operational state attributes

location? constraints?



Serious Cheese: Know Your Microbes

SERIOUS CHEESE / Say cheese! We recommend, review, and eat a lot of cheese.

JAKE LAHNE

PRINTER-FRIENDLY VERSION

Cabrales, a cheese ripened with blue *Penicillium* molds. [Photograph: jlastras on Flickr]

More

All About Cheese

Everything you need to know about eating and cooking with curds

The magic that is cheese only really needs four ingredients to happen: **milk, salt, rennet** (or some other coagulant, as I discussed earlier), and **microbes.** Like everyone, I used to be vaguely aware that there were "good" bacteria and molds that grew on and in cheese, and that's where my interest ended. But **there's a real variety of microbes that bring us the variety of cheeses we enjoy,** and they're worth knowing about. I would be a bad scientist if I didn't mention that, since I am no microbiologist, if you want all the details, you should peruse the Wikipedia articles I'll link to or consult your local library.

Many modern cheeses are made with preselected cultures, consisting of only a few types of microbe, but many traditional cheeses are inoculated using whey or other products from previous batches, meaning that they can be made with dozens of types of microbe, some highly unusual. **This microbial wealth is among the many reasons that traditional cheeses can be so much more complex than modern, controlled-inoculation cheeses.** Modern microbiology has yet to fully explain the role of all microbes in cheese-flavor and cheese-ripening, so the limited selection of controlled inoculation produces cheeses that may be less interesting.

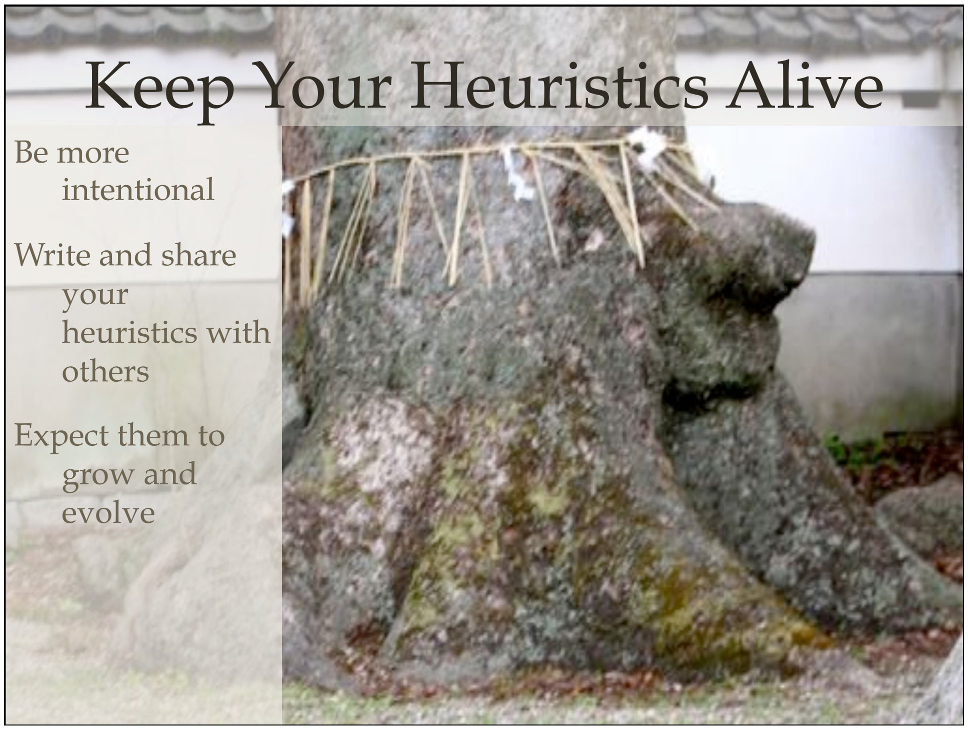
Cultivating Your Heuristics Requires Care and Attention

…notice what happens

when you apply a heuristic,

when you back up and try something else,

when you disagree on what to do next



Keep Your Heuristics Alive

Be more intentional

Write and share your heuristics with others

Expect them to grow and evolve

Credits & Acknowledgements

- ⋆ Erik Simmons encouraged me to read *Discussion of The Method.*
- ⋆ Richard Gabriel, a thinker and doer, critic of my work, and inspiration too.
- ⋆ Eric Evans makes me think deeply about design matters.
- ⋆ Mathias Verraes for sparking my heuristic exploration and continuing conversations about heuristics
- ⋆ Allen Wirfs-Brock photograph of Rebecca Wirfs-Brock at Haystack Rock.
- ⋆ Photographs were taken at DDD Europe 2018 of the workshop by the conference photographer and used with permission
- ⋆ All other photos taken by Rebecca Wirfs-Brock

Thank you!
rebecca@wirfs-brock.com
twitter: @rebeccawb
www.wirfs-brock.com