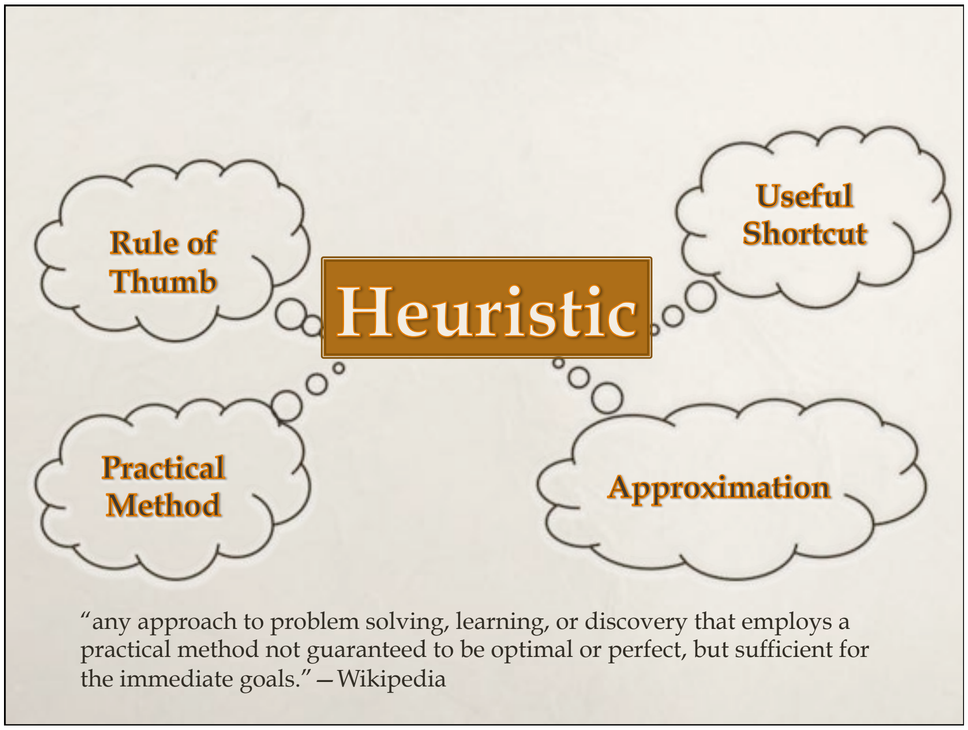
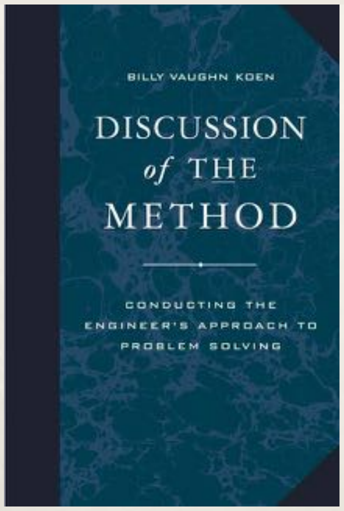
Growing Your Design Heuristics Toolkit

Rebecca Wirfs-Brock



"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



BILLY VAUGHN KOEN

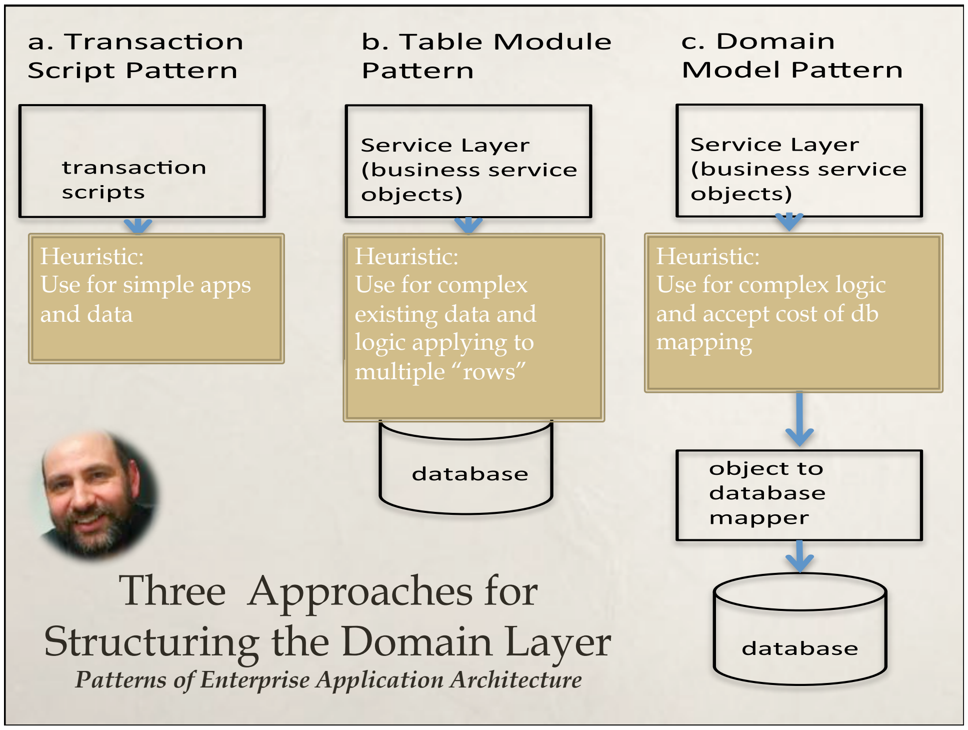
DISCUSSION of THE METHOD

CONDUCTING THE ENGINEER'S APPROACH TO PROBLEM SOLVING

"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

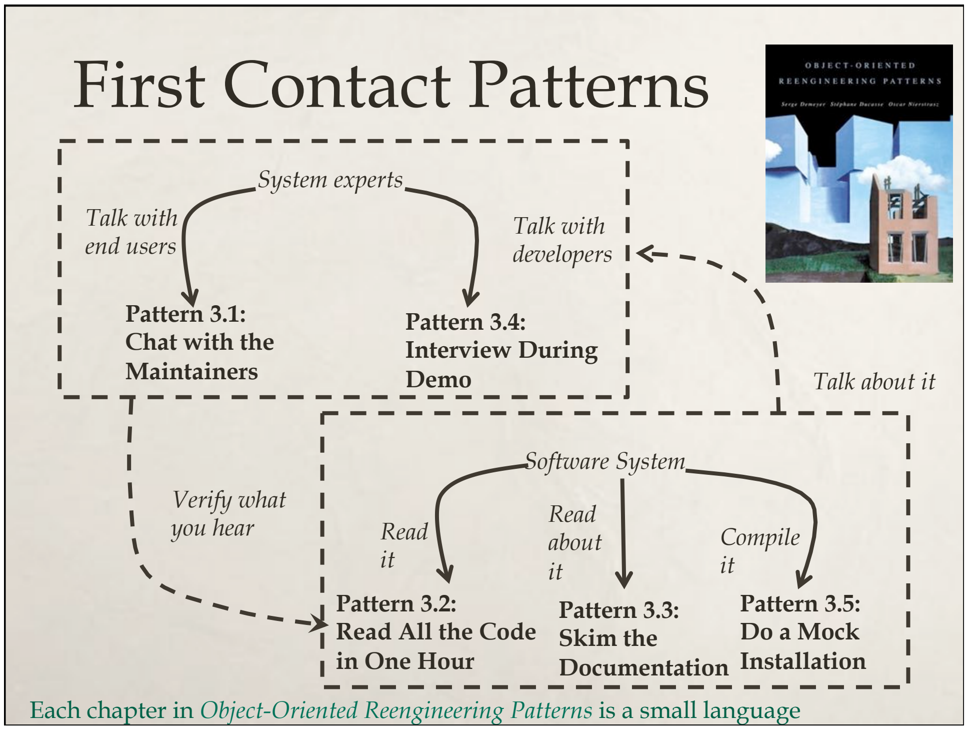
Heuristics to Solve a Design Problem


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

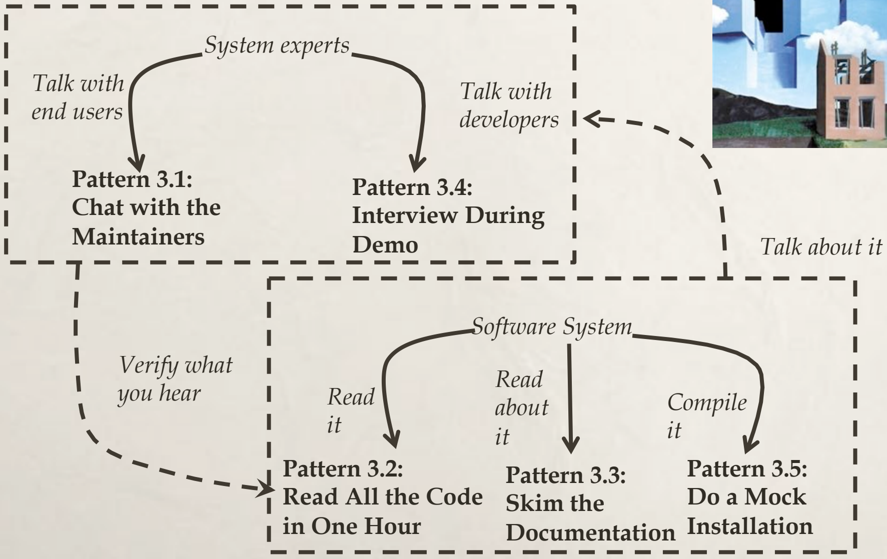
Three Approaches for Structuring the Domain Layer

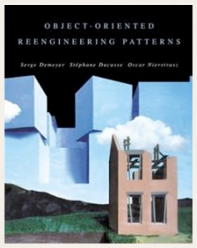
Patterns of Enterprise Application Architecture

Heuristics to Guide Use of Other Heuristics

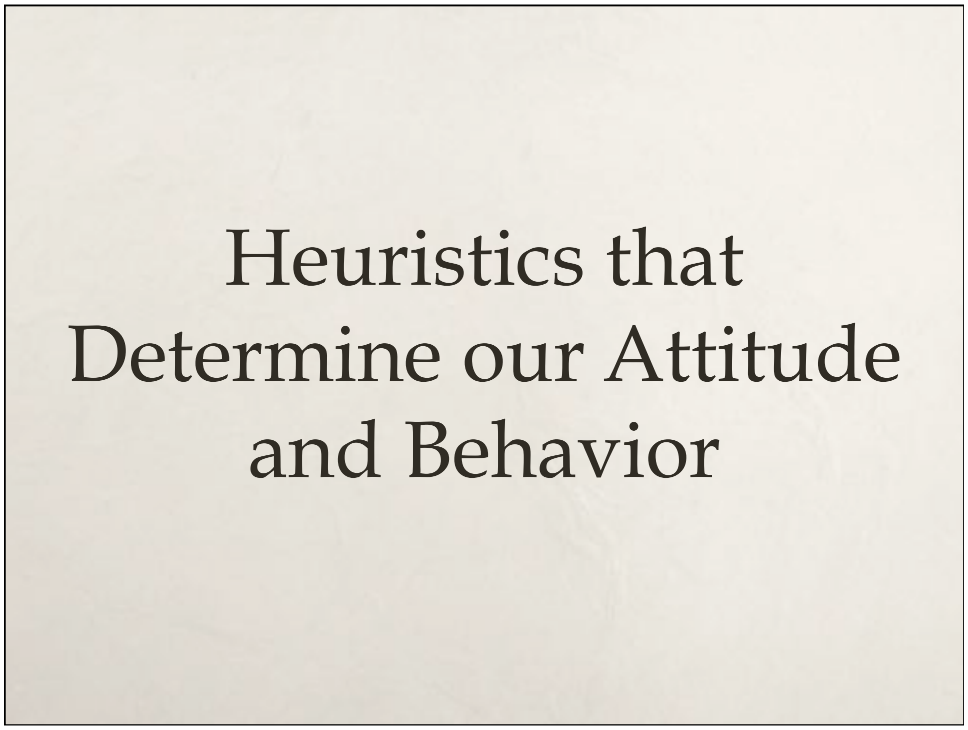


First Contact Patterns

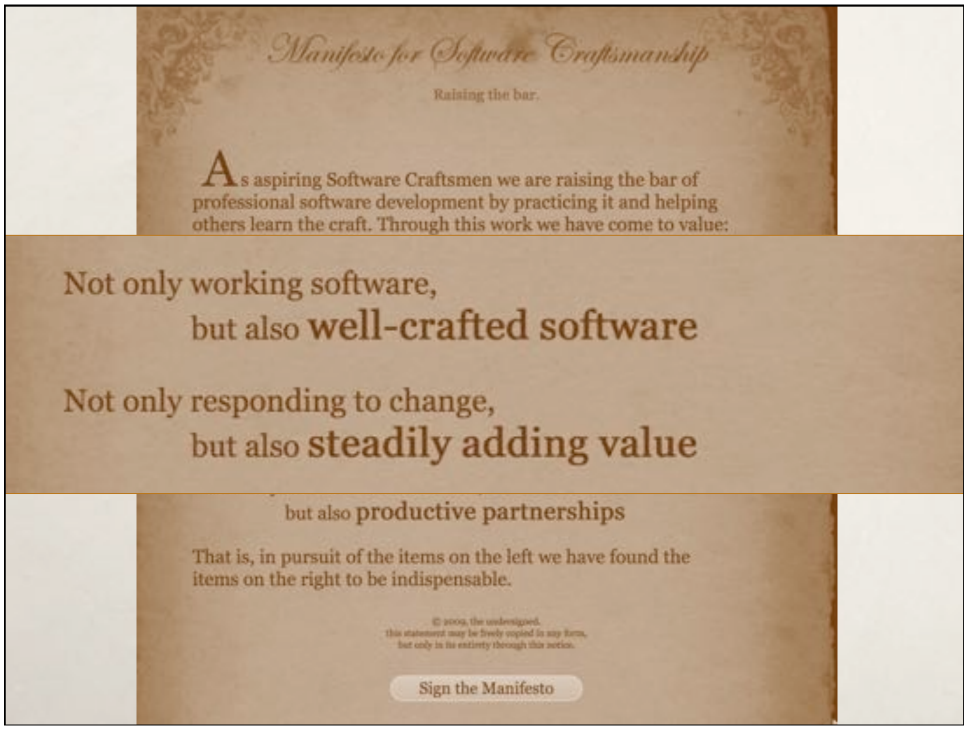

System experts
Talk with end users
Talk with developers
Pattern 3.1: Chat with the Maintainers
Pattern 3.4: Interview During Demo
Talk about it
Verify what you hear
Software System
Read it
Read about it
Compile it
Pattern 3.2: Read All the Code in One Hour
Pattern 3.3: Skim the Documentation
Pattern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



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



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**

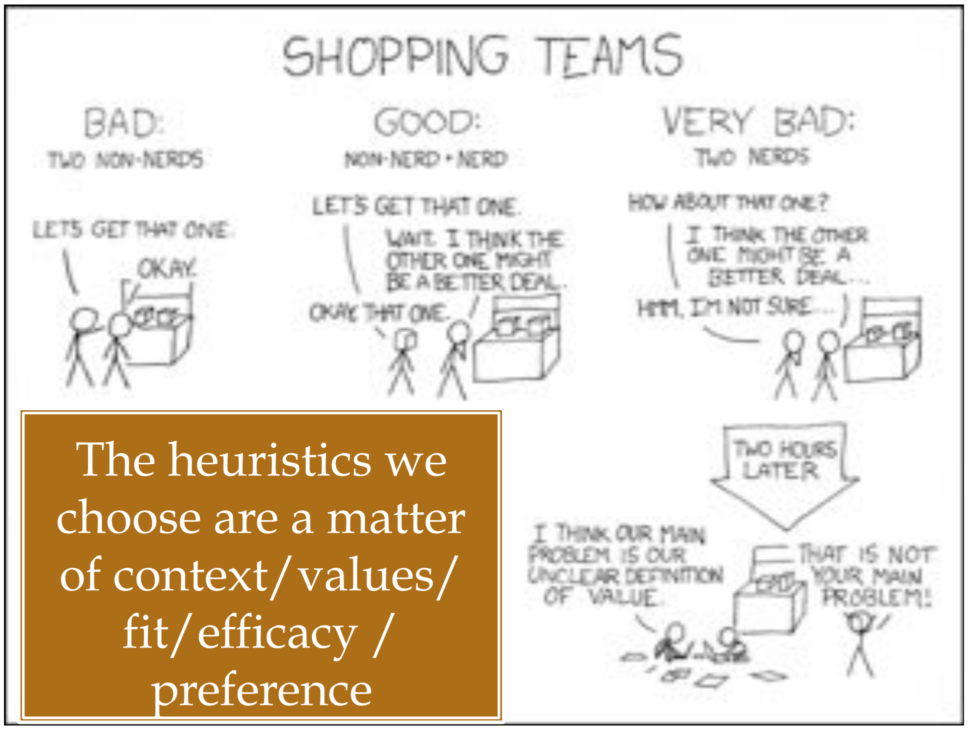
Not only responding to change,
but also **steadily adding value**

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.

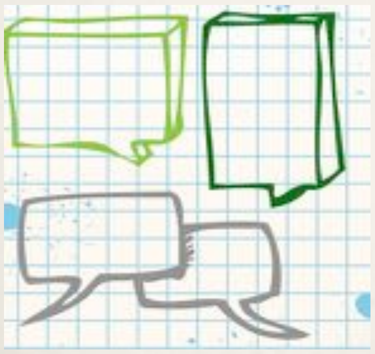
© 2009, the undersigned.
this statement may be freely copied in any form,
but only in its entirety through this notice.

Sign the Manifesto


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

Short Discussion



Share some cherished design heuristics with your neighbor

Share some cherished heuristics

What do typical heuristics look like?

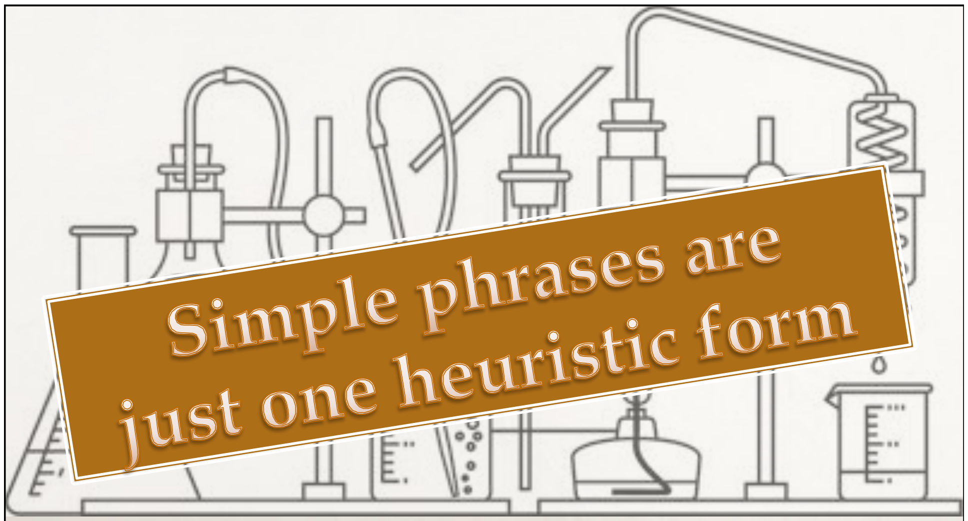
A Few General Engineering Heuristics by Billy

Solve problems by successive approximations.

Always give an answer.

Use feedback to stabilize your design.

Always give yourself a chance to retreat.



Simple phrases are just one heuristic form

There usually is a lot more behind any simple phrase:

Do this when … and ... unless ... and here's how...

Do this by first .... and then ... until ...

Context

In which situations
can I use this pattern?

Problem

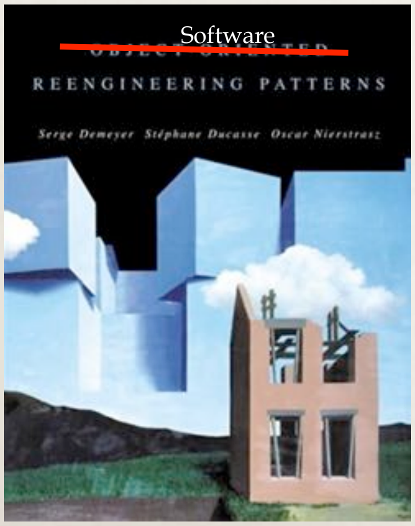
What does it try to solve?
What questions does it answer?

Solution

What can I
do that usually works?

patterns are another nicely "packaged" form

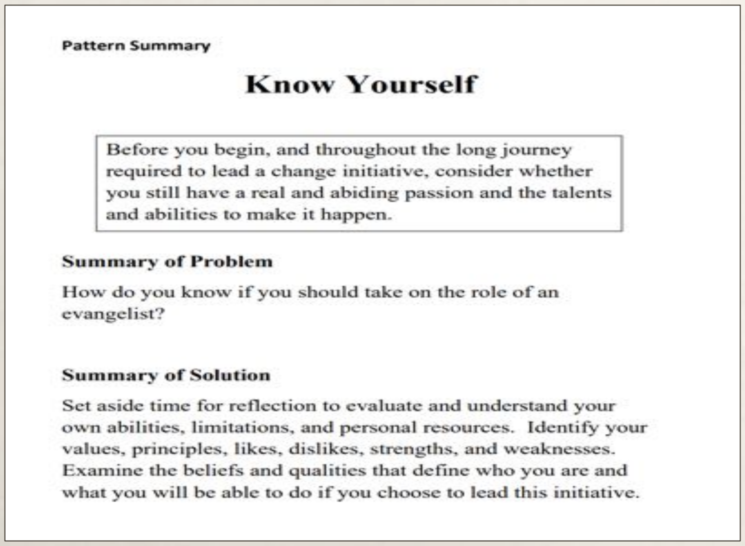
Pattern: Do a Mock Installation



Pattern: Do a Mock Installation

- **Intent:** Check whether you have the necessary artifacts available by installing the system and recompiling the code.
- **Problem:** How can you be sure that you will be able to (re)build the system?
- **Difficulties:**
  - The system is new to you, so you do not know which files you need.
  - The system may depend on libraries, frameworks, and patches, and you're uncertain you have the right versions available.
  - The system is large and complex, and the exact configuration under which the system is supposed to run is unclear.
  - Maintainers may answer these questions, or you may find answers in documentation, but you still must verify whether this information is complete.
- **Solution:** Try to install and build the system in a clean environment taking a limited amount of time (at most one day).
- **What next:** *Chat with the Maintainers* before you report your conclusions. When the build fails completely you may want to combine *Interview during Demo* with *Do a Mock Installation*

Heuristic Gists*



Pattern Summary

Know Yourself

Before you begin, and throughout the long journey required to lead a change initiative, consider whether you still have a real and abiding passion and the talents and abilities to make it happen.

Summary of Problem

How do you know if you should take on the role of an evangelist?

Summary of Solution

Set aside time for reflection to evaluate and understand your own abilities, limitations, and personal resources. Identify your values, principles, likes, dislikes, strengths, and weaknesses. Examine the beliefs and qualities that define who you are and what you will be able to do if you choose to lead this initiative.

*Similar to pattern thumbnails…example here is from Fearless Change patterns, but you can write up heuristics this way, too

Another Option For a First Cut:
Question, Heuristic, Example (QHE) Cards

Q. When should I generate a different event?

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

Heuristic

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

Write a Heuristic on a QHE Card. Include the question, the heuristic (answer), and examples

Short exercise

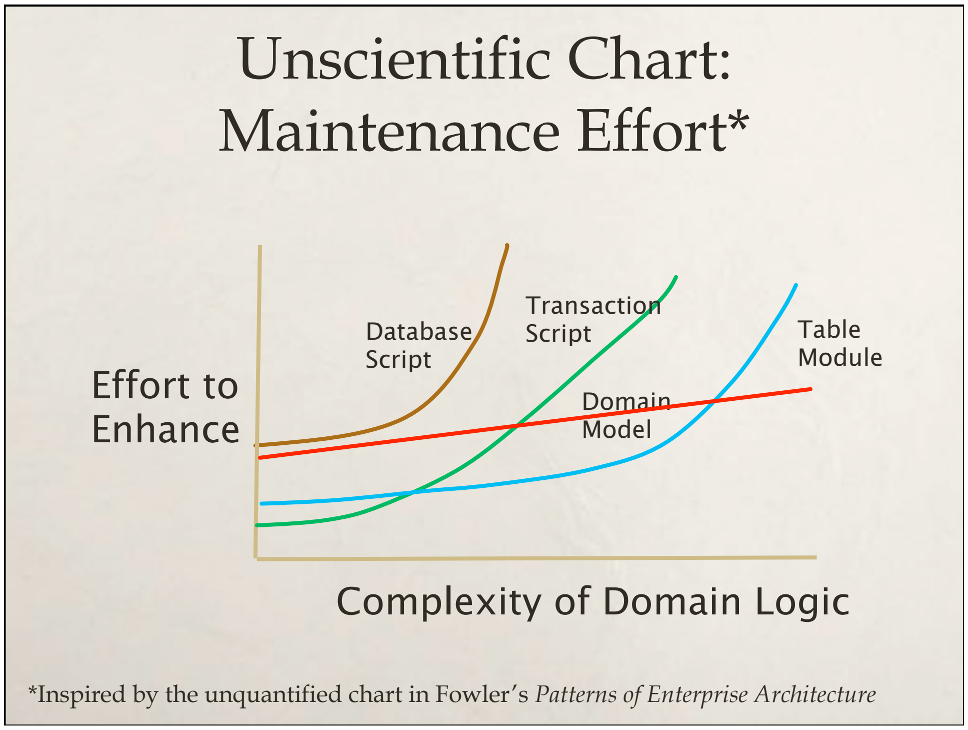
Heuristics Need to be Challenged

© Can Stock Photo / 4774344sean

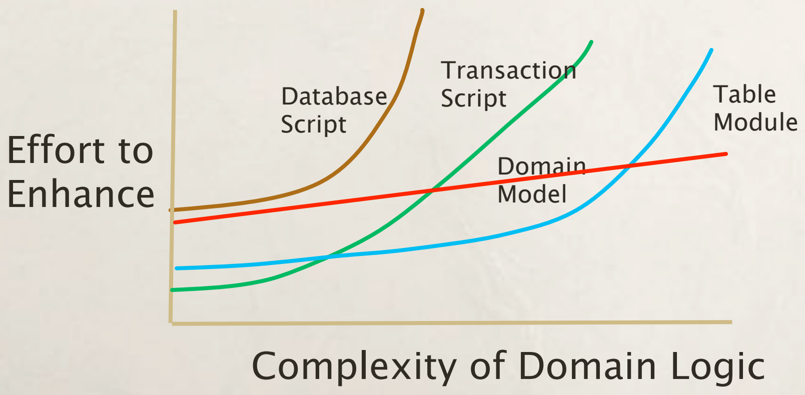
Heuristics:
3 Ways to Structure a Domain Layer

Patterns of Enterprise Application Architecture

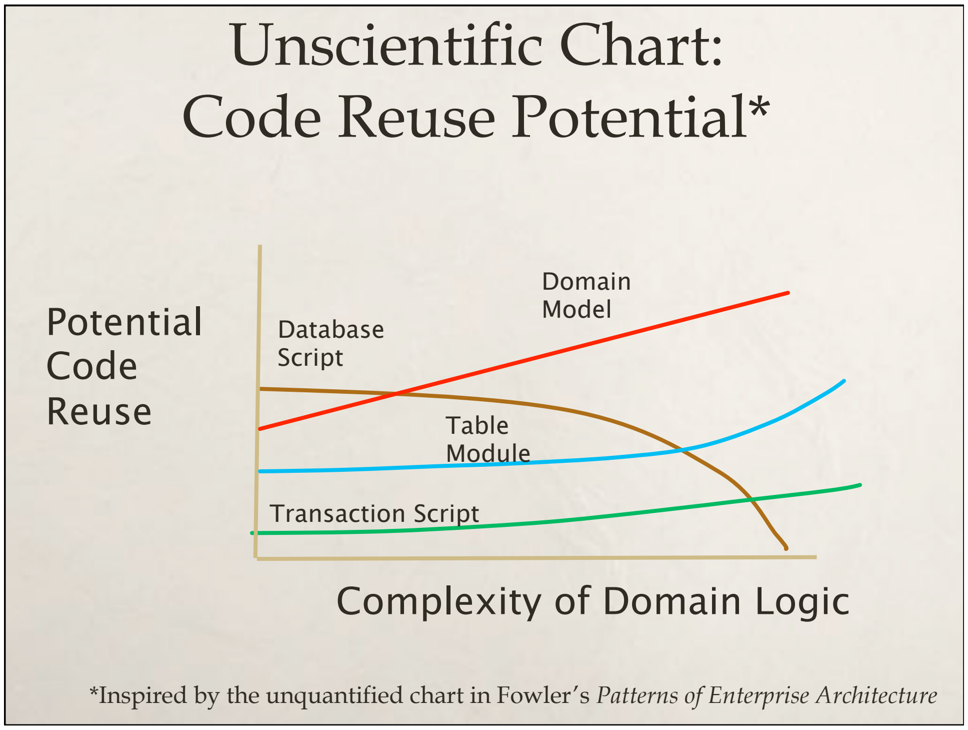
What about stored proceedures, rules engines, "simple" domain models, or functional programming solutions?



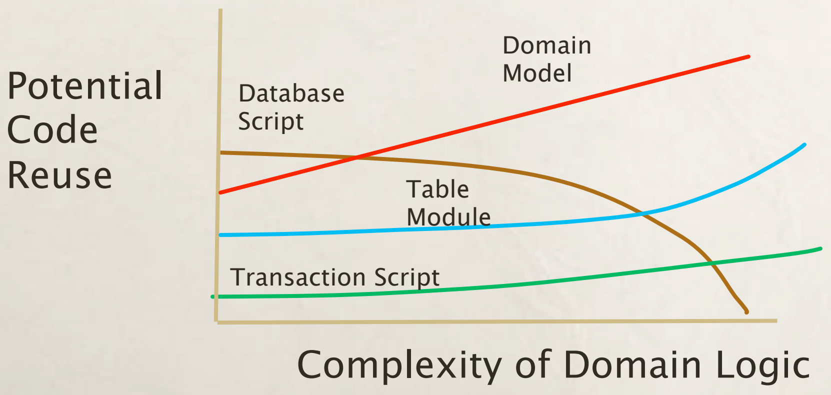
Unscientific Chart: Maintenance Effort*



*Inspired by the unquantified chart in Fowler's *Patterns of Enterprise Architecture*



Unscientific Chart: Code Reuse Potential*



*Inspired by the unquantified chart in Fowler's *Patterns of Enterprise Architecture*

Heuristics:
3 Ways to Structure a Domain Layer

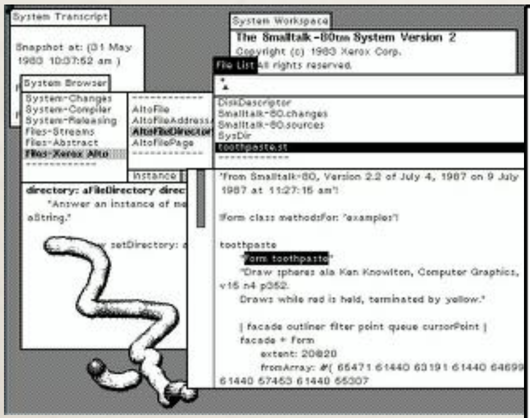
Patterns of Enterprise Application Architecture

But Martin, what about CQRS archi ures or ervice architectures?

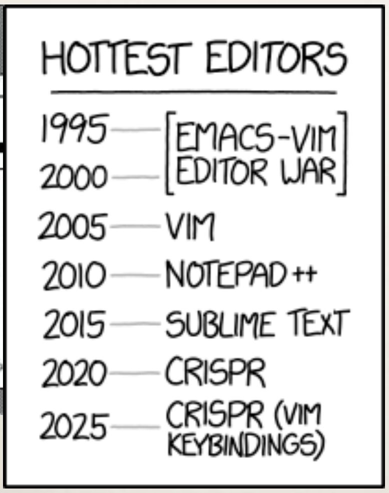
NO FAIR

…but don't judge an older system (or its designer) based on today's heuristics.

Our State of The Art *(SOTA)*
According to Vaughn Koen



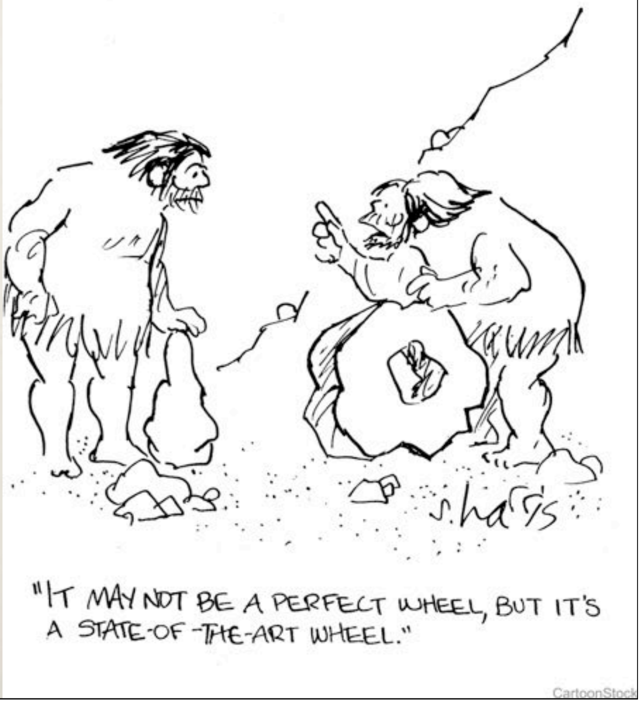
```
Snapshot at: (31 May
1983 10:37:52 am )

The Smalltalk-80tm System Version 2
Copyright (c) 1983 Xerox Corp.
All rights reserved.

System-Changes
System-Compiler
System-Releasing
Files-Streams
Files-Abstract
Files-Xerox Alto
------------

------------
AltoFile
AltoFileAddress
AltoFileDirector
AltoFilePage
------------

DiskDescriptor
Smalltalk-80.changes
Smalltalk-80.sources
SysDir
toothpaste.st
------------

directory: aFileDirectory direc
    "Answer an instance of me
aString."

        setDirectory: a

'From Smalltalk-80, Version 2.2 of July 4, 1987 on 9 July
1987 at 11:27:15 am'!

!Form class methodsFor: 'examples'!

toothpaste
    "Form toothpaste"
    "Draw spheres ala Ken Knowlton, Computer Graphics,
v15 n4 p352.
    Draws while red is held, terminated by yellow."

    | facade outliner filter point queue cursorPoint |
    facade ← Form
        extent: 20@20
        fromArray: #( 65471 61440 63191 61440 64699
61440 57463 61440 65307
```



- Sometimes, even useful ones fade away

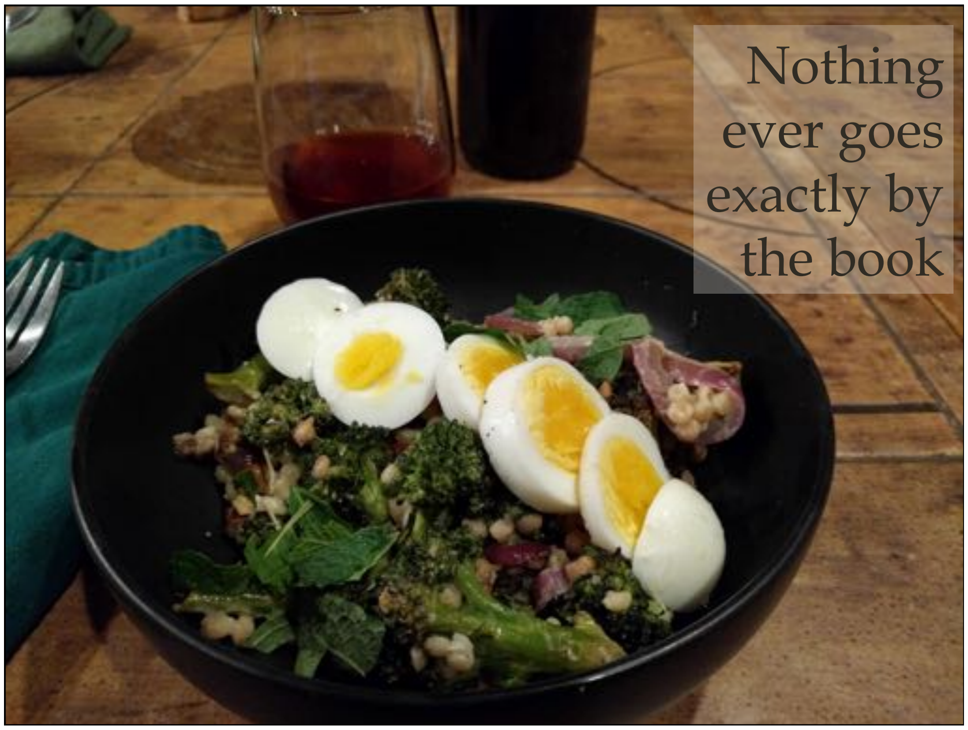
https://xkcd.com/1823/

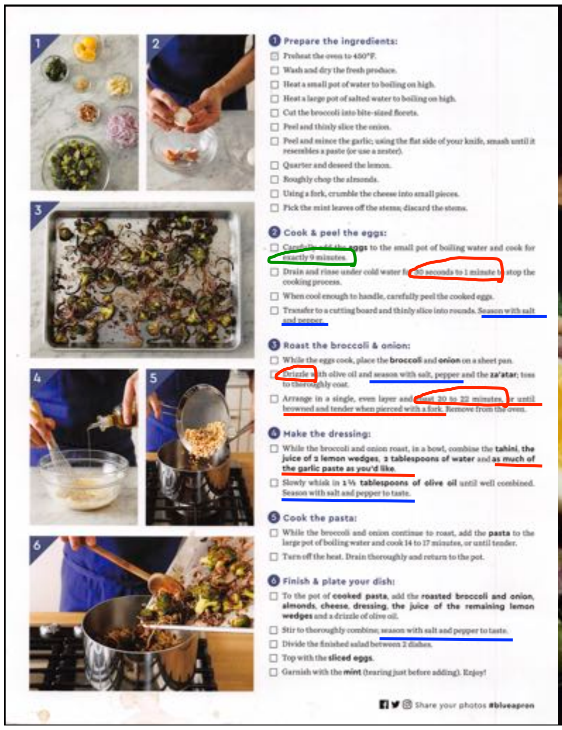
Our state-of-the-art is constantly progressing



"IT MAY NOT BE A PERFECT WHEEL, BUT IT'S A STATE-OF-THE-ART WHEEL."

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





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.

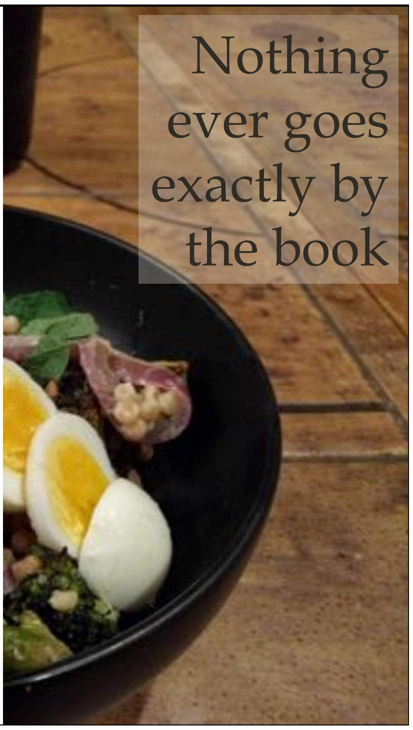
5 Cook the pasta:

- ☐ 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.

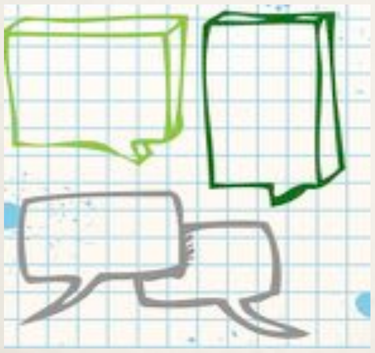
6 Finish & plate your dish:

- ☐ 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



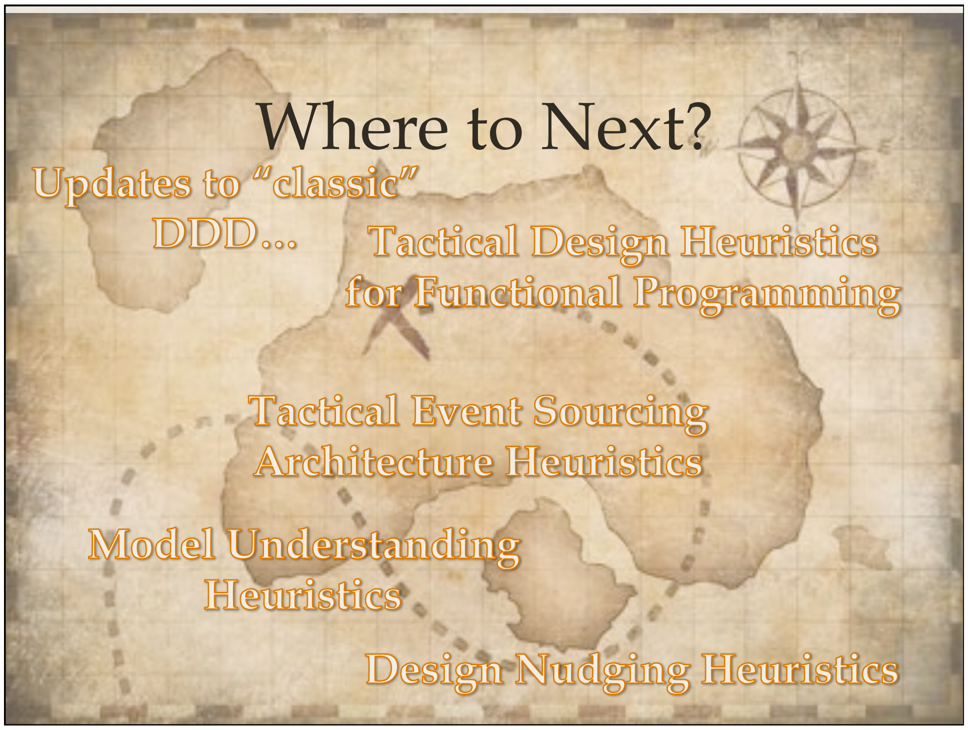
How have your heuristics have evolved?

Short discussion

Techniques for Actively Cultivating Your Heuristics

Map out What You Know

Map out Your Interests



Where to Next?

Updates to "classic" DDD…

Tactical Design Heuristics for Functional Programming

Tactical Event Sourcing Architecture Heuristics

Model Understanding Heuristics

Design Nudging Heuristics

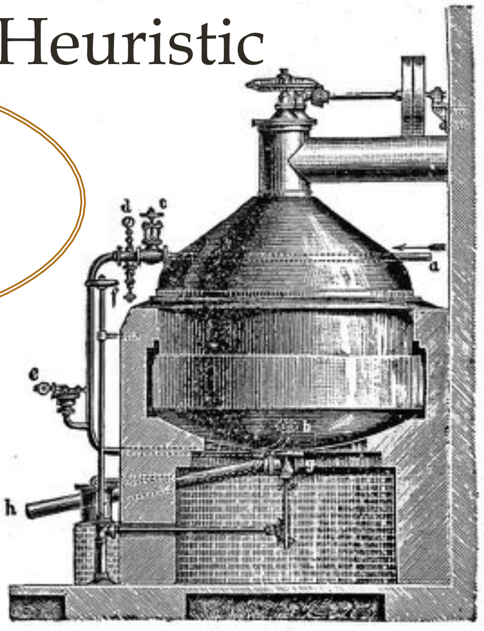
1. Compare your preferred heuristics with others'

"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.



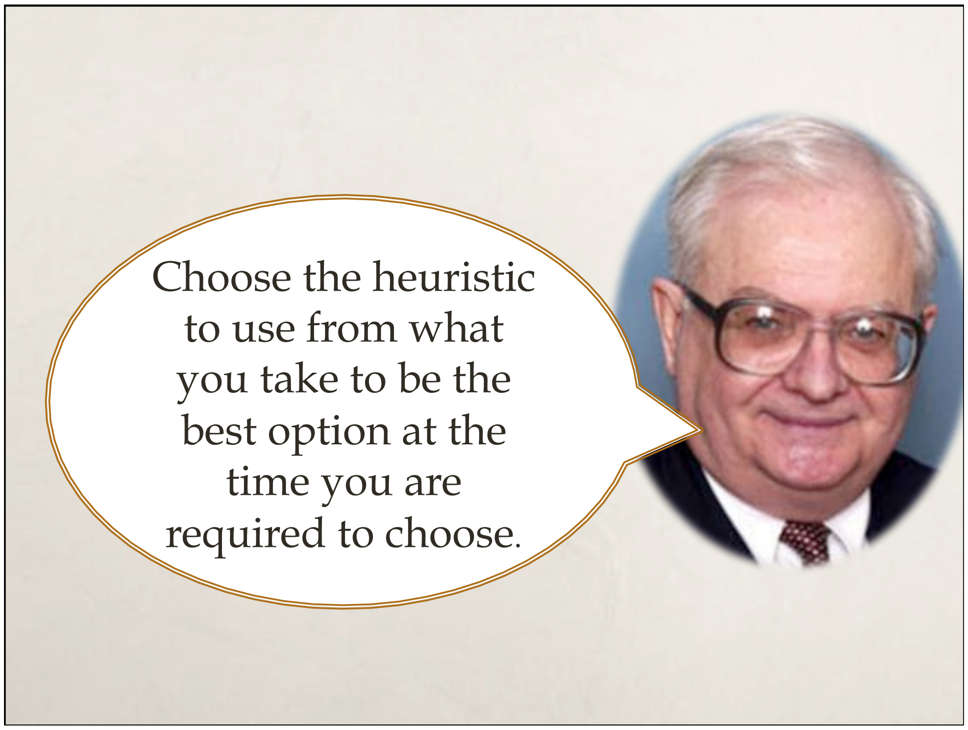
© Can Stock Photo / gameover

My Debate with Paul

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

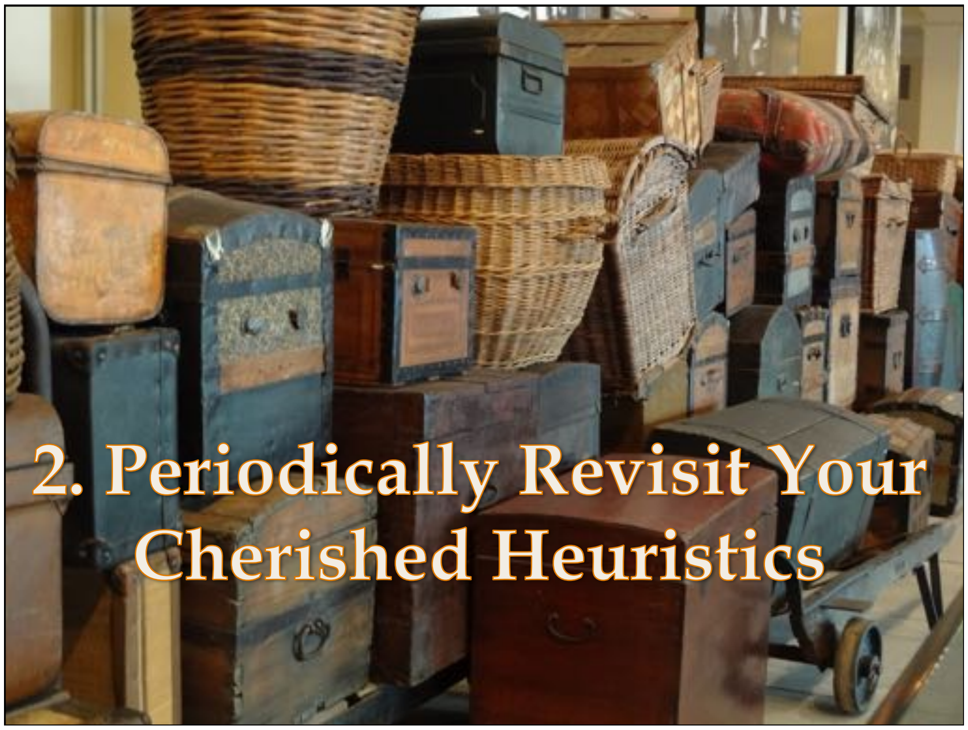
And, use transaction scripts for really simple stuff that isn't going to change much.

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


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

the reality…

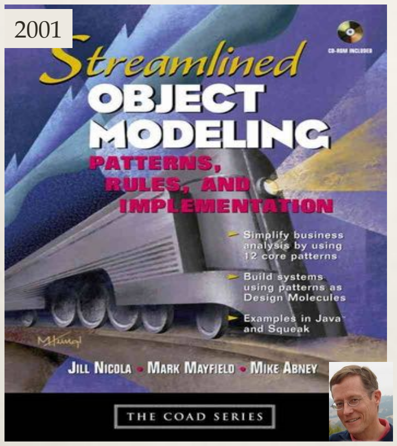
“All we can do is the best we can do.”
—David Axelrod



2. Periodically Revisit Your Cherished Heuristics

Heuristic: By characterizing a domain entity's attributes you can understand/find/identify needed system behaviors

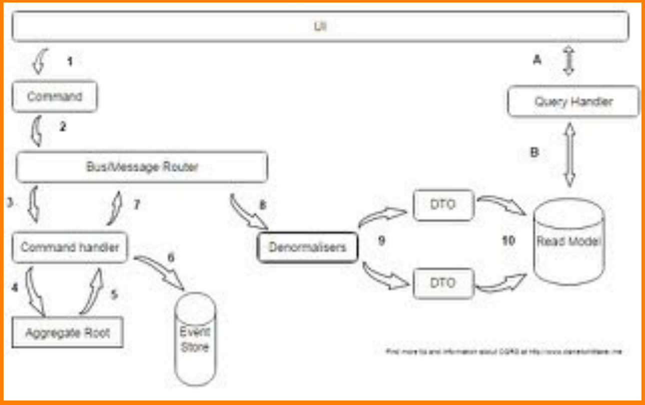
- *Descriptive Attributes* reflect a domain's properties (not identity).
- *Time-dependent attributes* Where aintaining a history of past values is important.
- *Lifecycle state attributes* Some entities go through a one-way lifecycle, from initial to final state.
- *Operational state* Some entities 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

My Heuristics for Validating Data

- Perform simple edits (syntactic) in browser code
- Don't universally 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 layer 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…

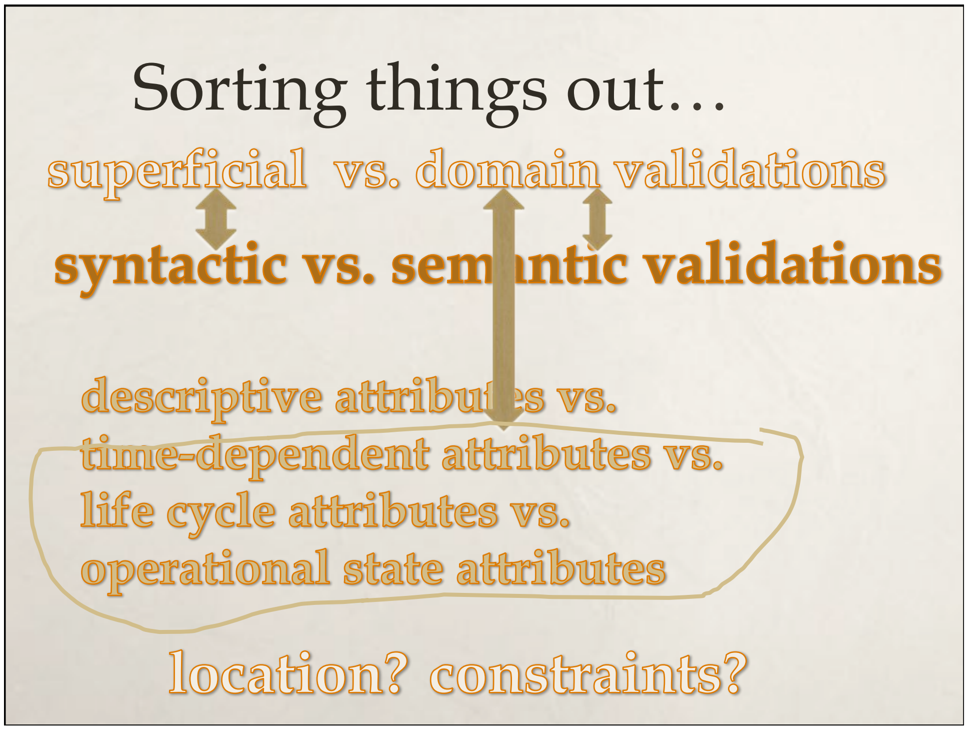
superficial vs. domain validations

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



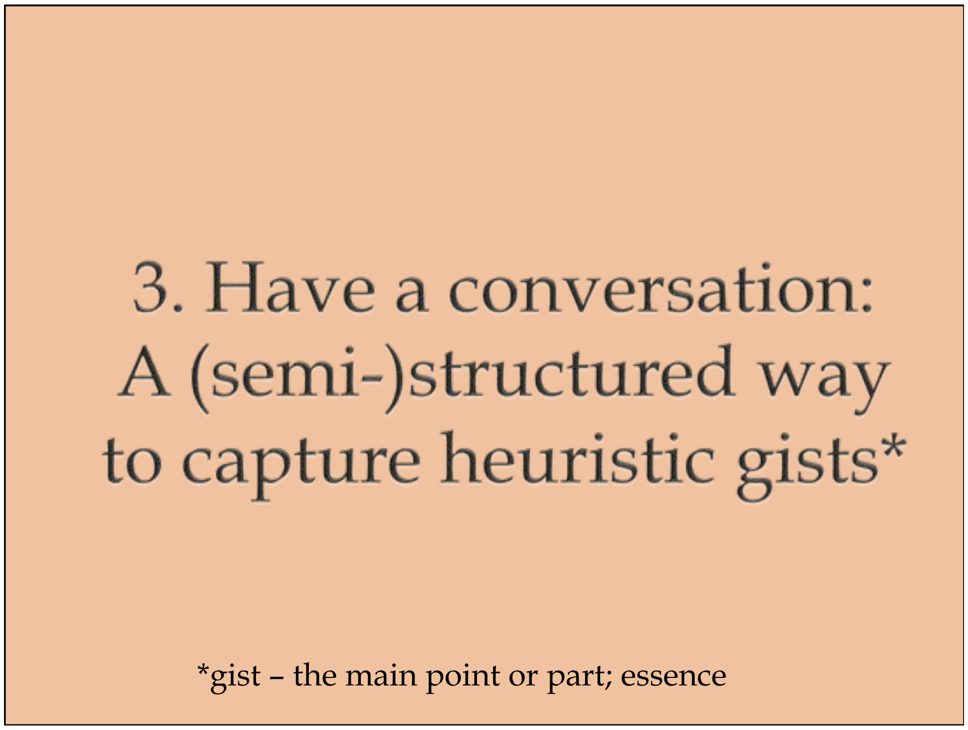
Sorting things out…

superficial vs. domain validations

syntactic vs. semantic validations

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

location? constraints?



3. Have a conversation: A (semi-)structured way to capture heuristic gists*

*gist – the main point or part; essence

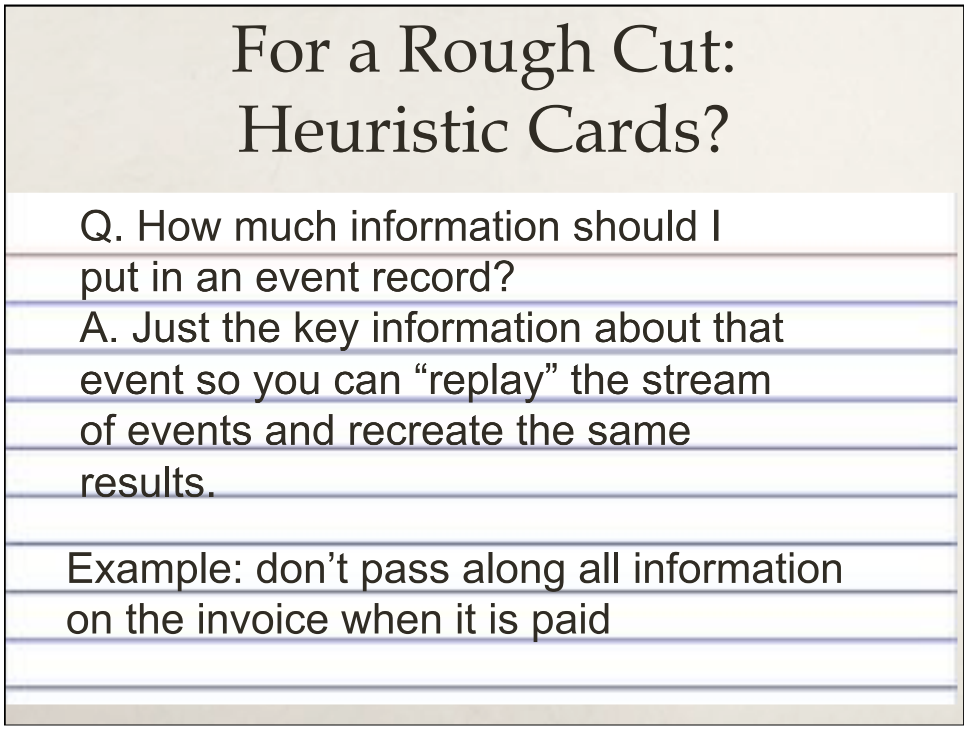
Distillation Conversations

A 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.**

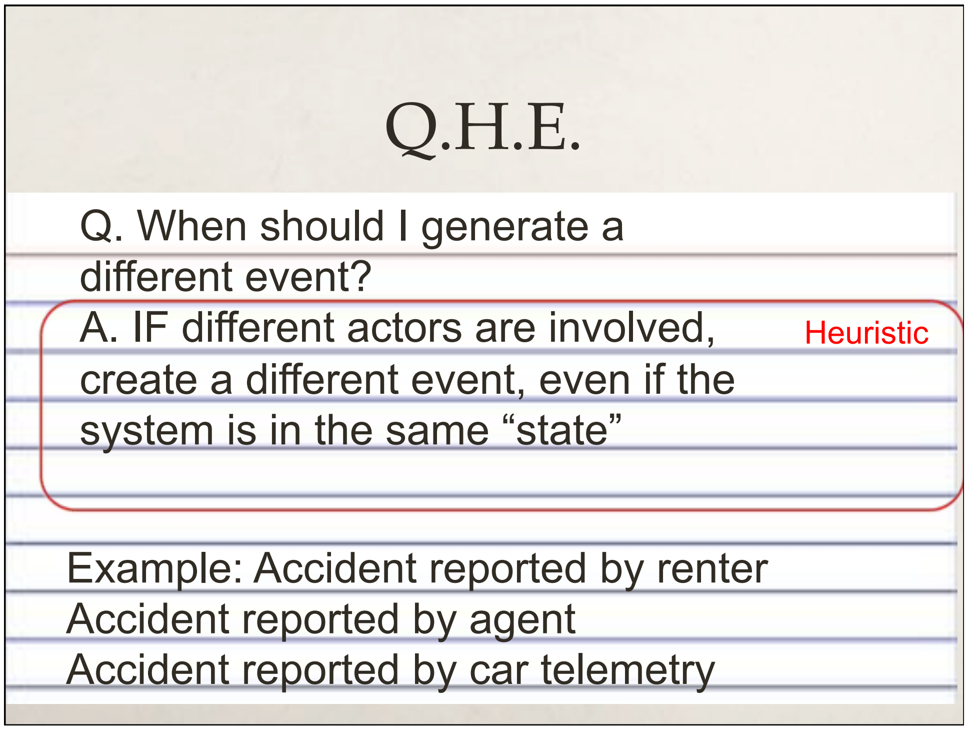
The event is "a reservation has been made" or "service has been scheduled"



For a Rough Cut: Heuristic Cards?

Q. How much information should I put in an event record?
A. Just the key information about that event so you can "replay" the stream of events and recreate the same results.

Example: don't pass along all information on the invoice when it is paid



Q.H.E.

Q. When should I generate a different event?
A. IF different actors are involved, create a different event, even if the system is in the same "state"

Heuristic

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

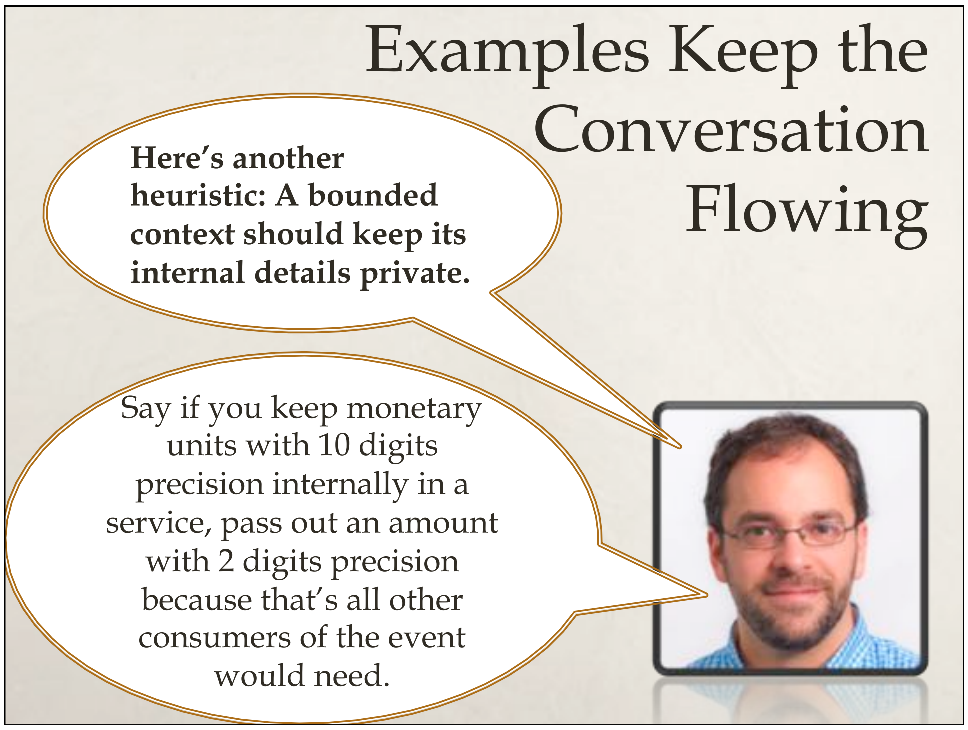
Q.H.E.

Question: How many events should you generate?
Heuristic: if there are different behaviors downstream, then there are different events generated from the same process.

Example: Car returned: Car returned event
Car mileage recorded event

Have a Structured Conversation and Distill Some Heuristics

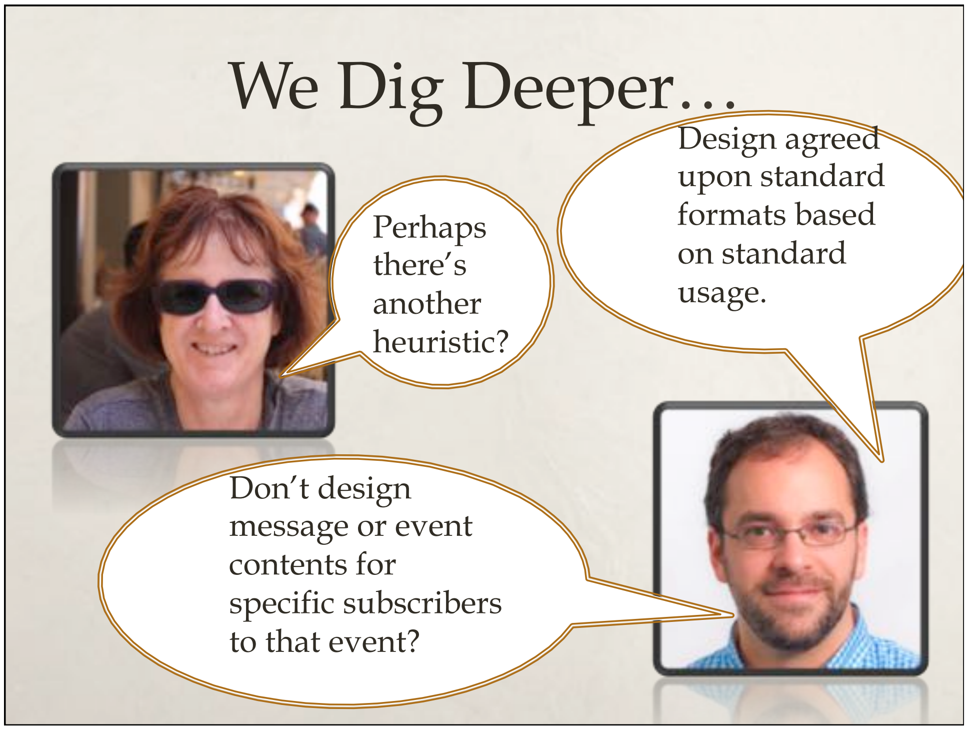
How do you approach doing…?



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.

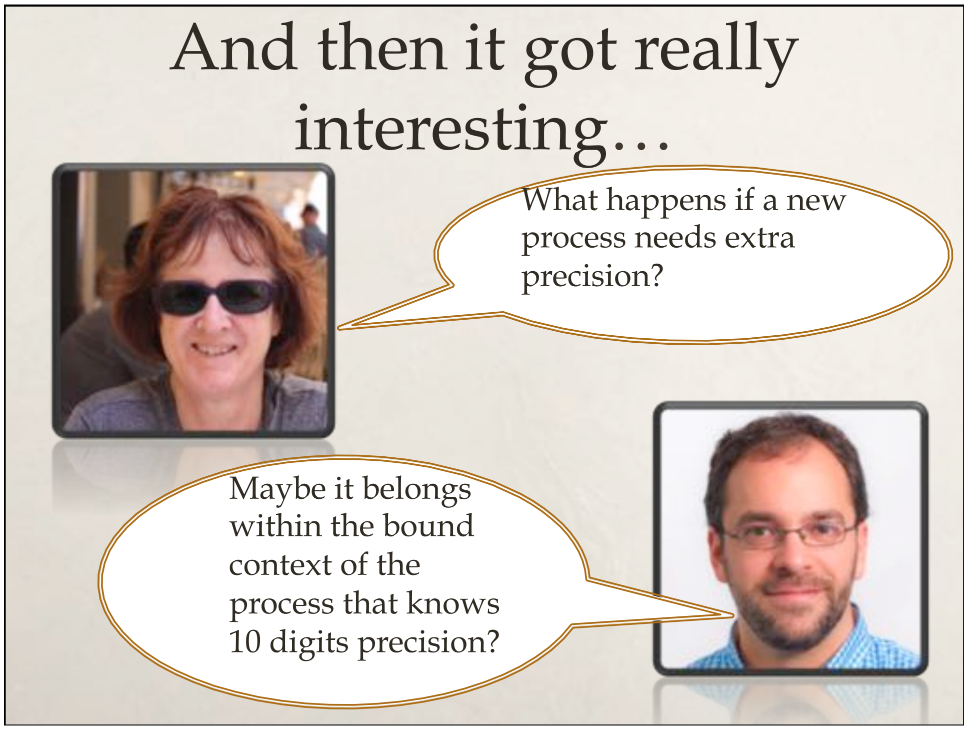


We Dig Deeper…

Perhaps there's another heuristic?

Design agreed upon standard formats based on standard usage.

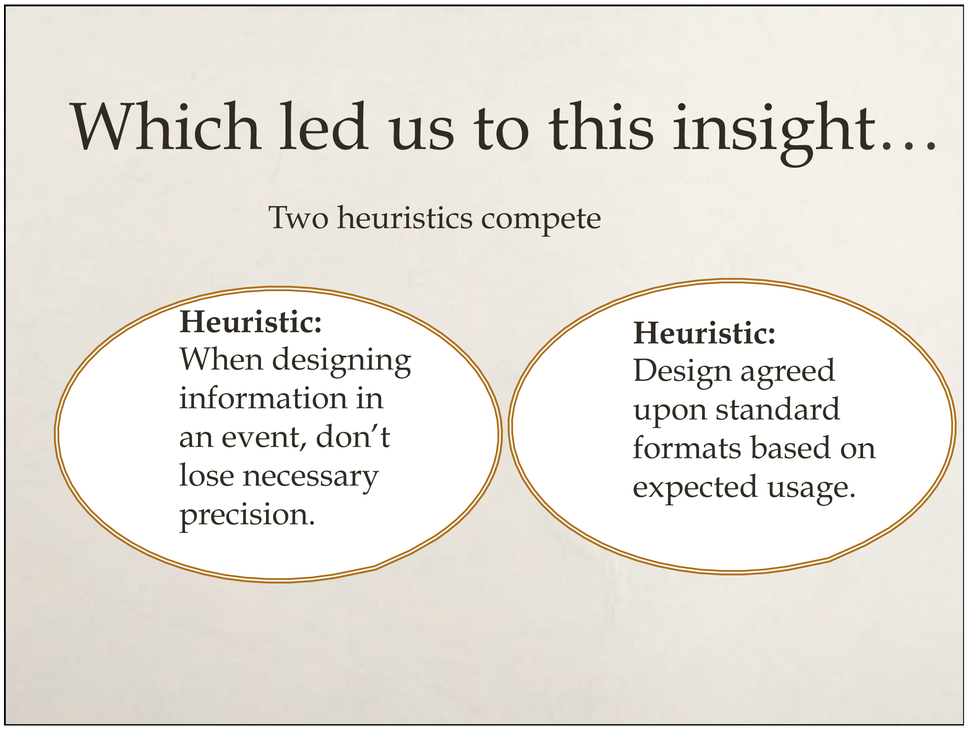
Don't design message or event contents for specific subscribers to that event?



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…

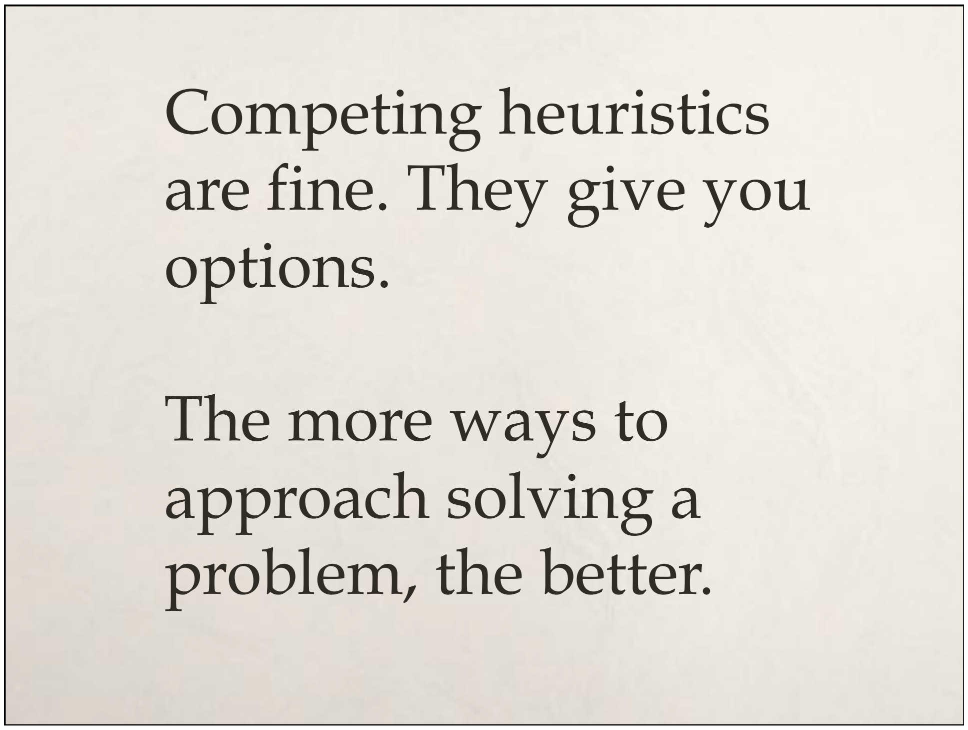
Two heuristics compete

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

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

Heuristics May Conflict…

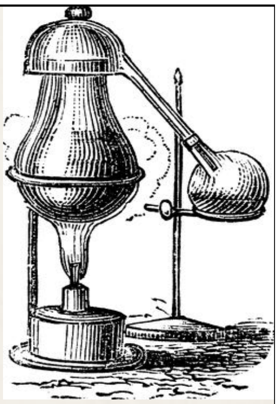
and still be useful



Competing heuristics are fine. They give you options.

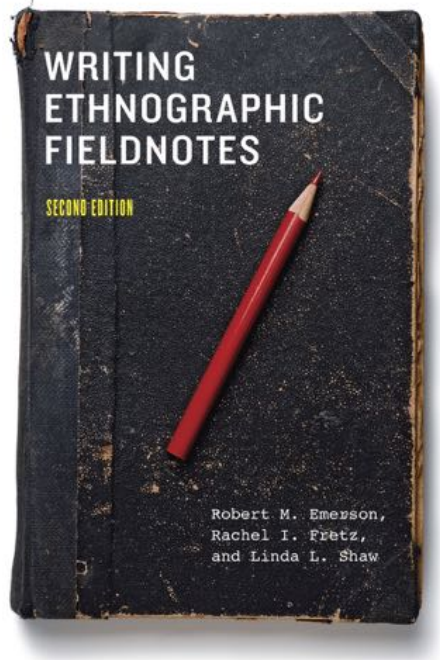
The more ways to approach solving a problem, the better.

Distiller Advice



* Listen
* Let the conversation wander where the person you are trying to glean knowledge from wants 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.

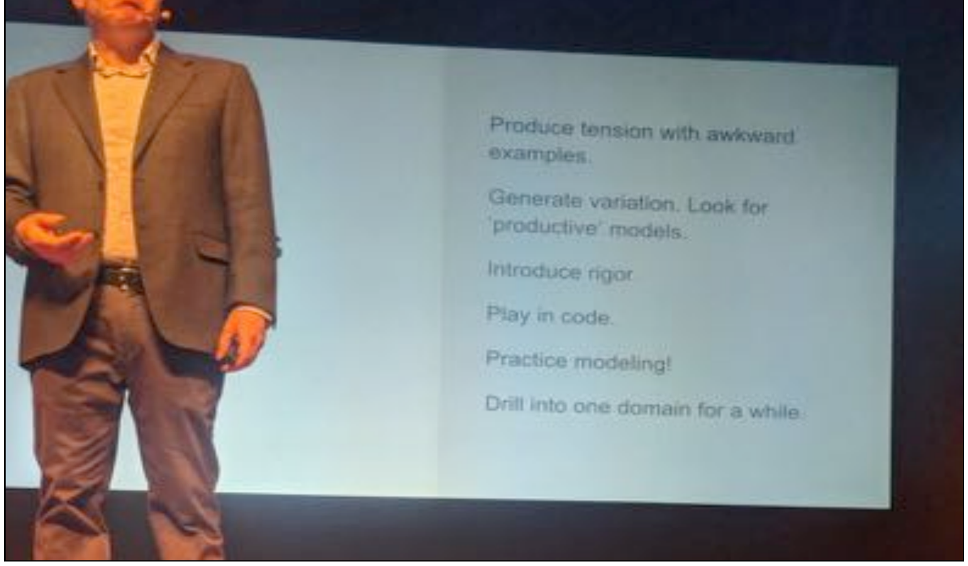
4. Take notes of how you actually work



Radical idea: Take notes on how you design at work

When?
after work
times when you have half an hour...

5. Distill What You Hear at Conferences



Produce tension with awkward examples.

Generate variation. Look for 'productive' models.

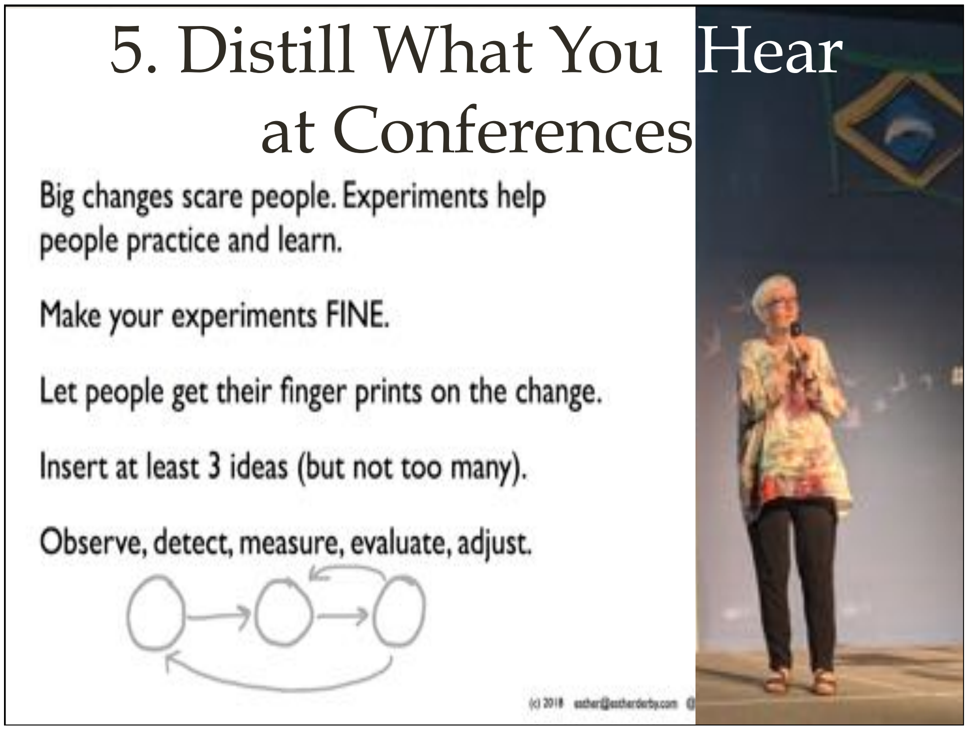
Introduce rigor.

Play in code.

Practice modeling!

Drill into one domain for a while.

5. Distill What You Hear at Conferences



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

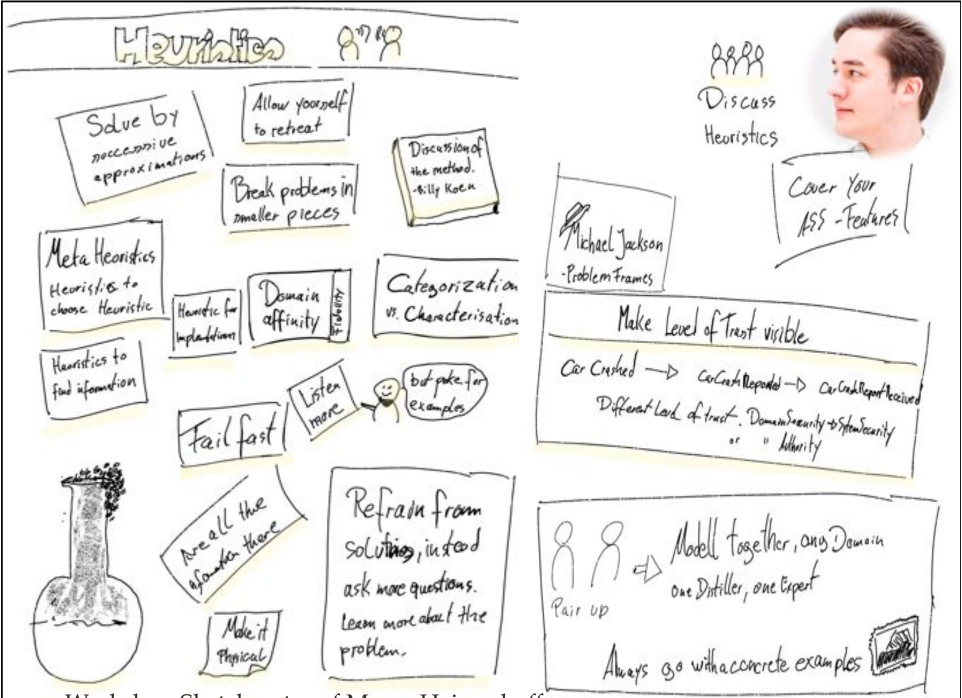
Make your experiments FINE.

Let people get their finger prints on the change.

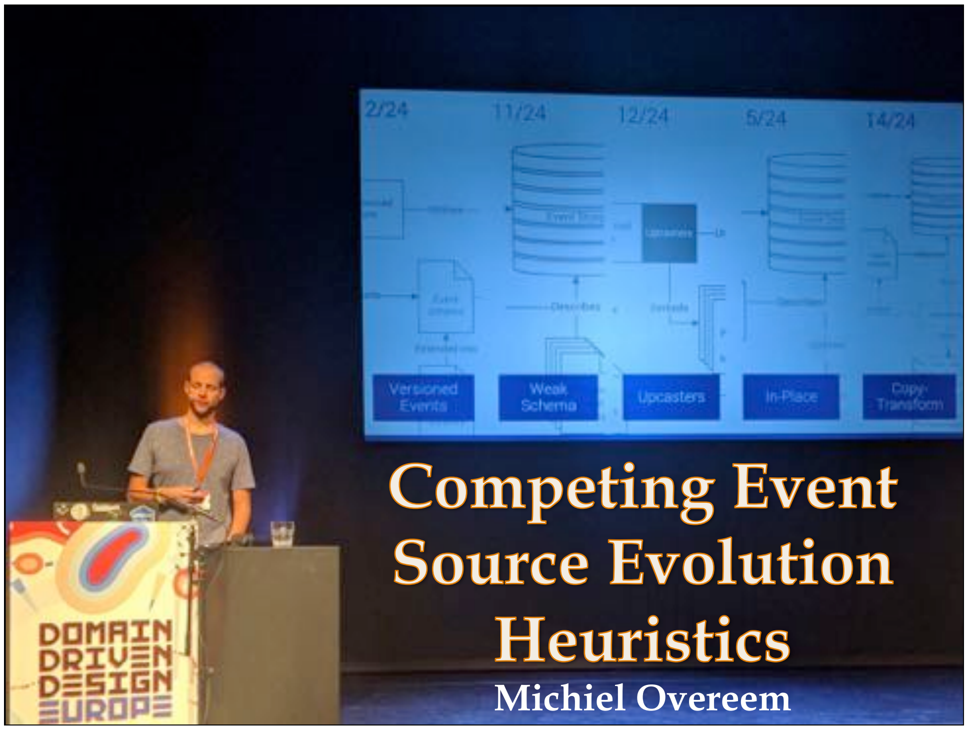
Insert at least 3 ideas (but not too many).

Observe, detect, measure, evaluate, adjust.

(c) 2018 esther@estherderby.com


Heuristics
Solve by successive approximations
Allow yourself to retreat
Discussion of the method. -Billy Koen
Break problems in smaller pieces
Meta Heuristics
Heuristics to choose Heuristic
Heuristic for [illegible]
Domain affinity
Fidelity
Categorization vs. Characterisation
Heuristics to find information
Listen more
but poke for examples
Fail fast
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
Modell together, any Domain
One Distiller, one Expert
Pair up
Always go with a concrete examples

Workshop Sketch notes of Marco Heimeshoff



Competing Event Source Evolution Heuristics

Michiel Overeem

Be a Heuristic Champion

6 Advocacy

1. Books, blogs, case studies, critiques
2. Reference apps
3. Journaling, design logs..

Advocacy: Journaling
Describe Your Design Values & Principles

Guidelines:

Prefer a rich domain model

Aggregate roots should not directly communicate with each other

Conventions:

Common service interfaces/capabilities

Extension points configured by…

When you break the rules

… and why

75

Advocacy: Journaling
Document Design Decisions
One option I like*

1-2 pages describing a set of forces forces & a single decision in response

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

Decisions worth documenting

Spent lots of time on

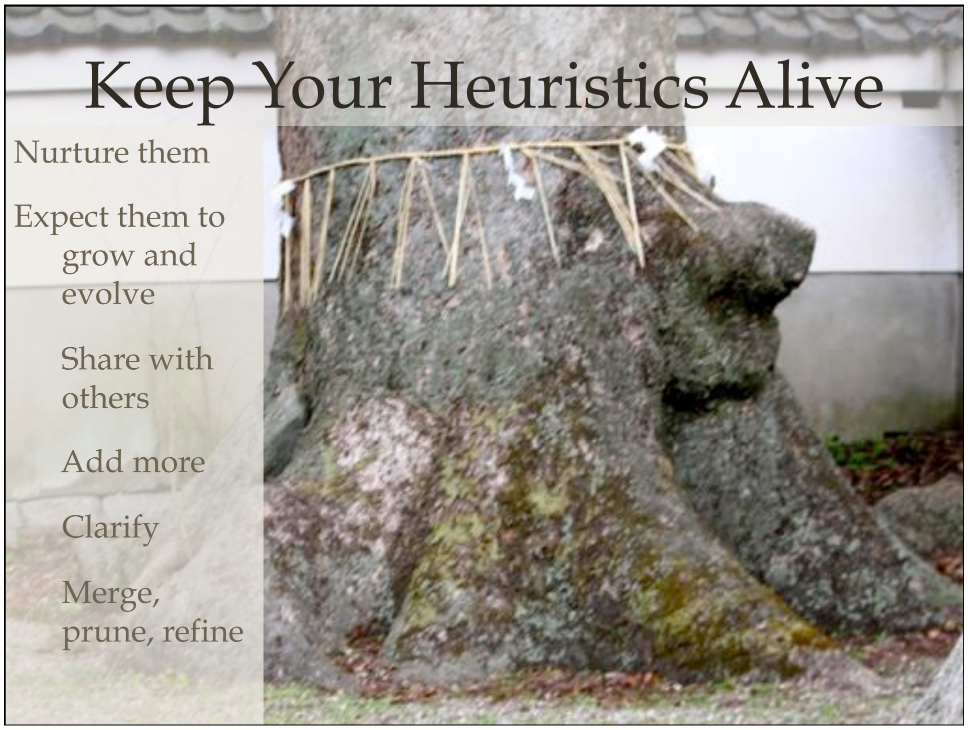
Critical to achieving a requirement

Confusing at first

Widespread impact

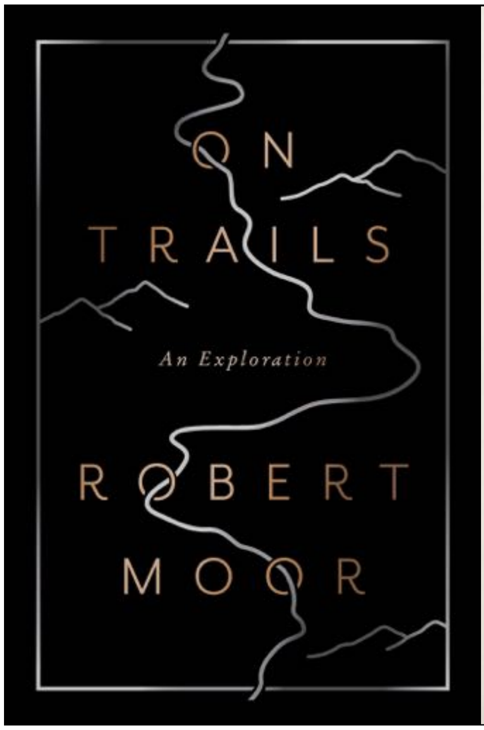
Difficult to undo

*Useful for recorded decisions that have a "lifecycle". 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

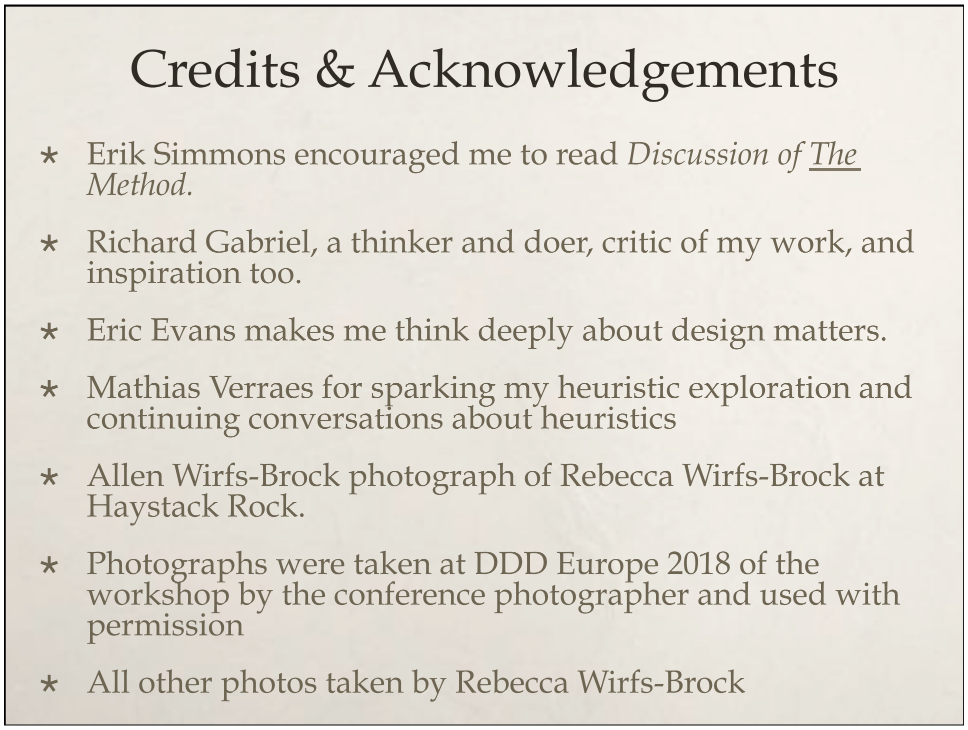


Keep Your Heuristics Alive

- Nurture them
- Expect them to grow and evolve
  - Share with others
  - Add more
  - Clarify
  - Merge, prune, refine



"An explorer finds a worthwhile destination; then every walker who follows that trail makes it a little better. Ant trails, game paths, ancient ways, modern hiking trails—they all continually adapt to the aims of their walkers."



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