

My Silver Toolbox

Simon Brown

Will Chaparro

George Fairbanks

Ariadna Font

Gail Harris

Michael Keeling

Eric Willeke

Silver Toolbox





“I can’t give you a
silver bullet.

But I can give you
a silver tool box.”

- Mel Rosso-Llopart,
Carnegie Mellon University

APRIL 1987

COMPUTER

No Silver Bullet

*Fred Brooks
on
Avoiding Horrors
in the Software
Engineering
Process*



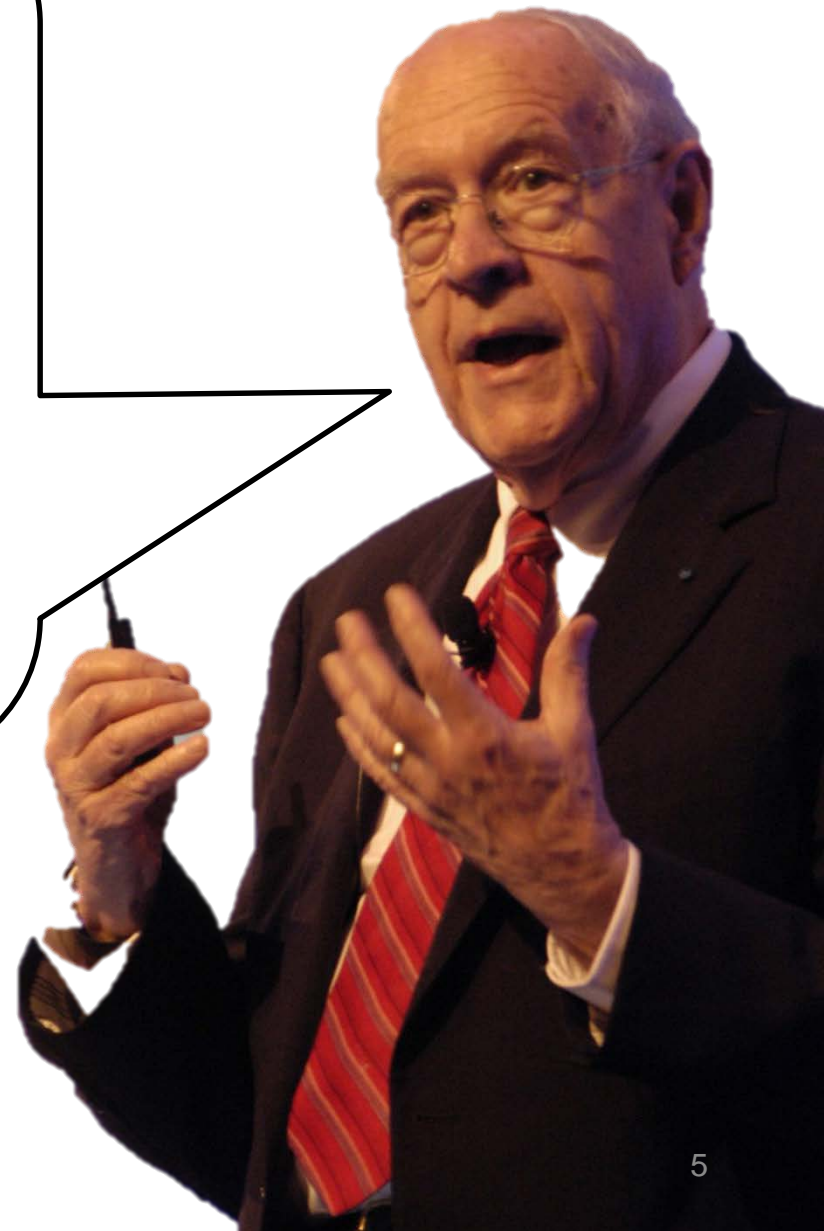
THE COMPUTER SOCIETY
OF THE IEEE



THE INSTITUTE OF ELECTRICAL AND
ELECTRONICS ENGINEERS, INC.

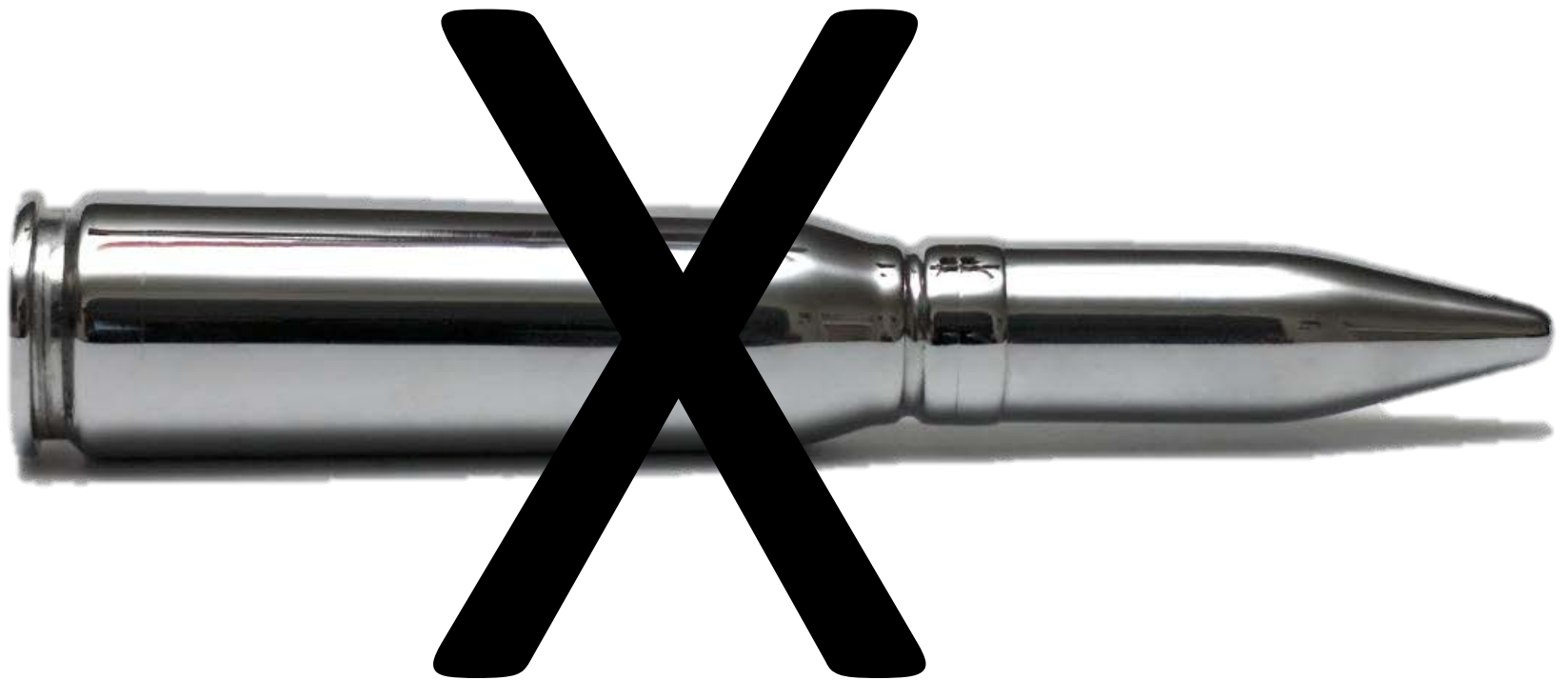
“There is no single development, in either technology or management technique, which by itself promises even one order of magnitude improvement within a decade in productivity, in reliability, in simplicity.”

Fred Brooks, “No Silver Bullet — Essence and Accidents of Software Engineering”, 1986



Complexity's a Beast





No Silver.... Denied!







What's
in your
silver
toolbox?

20 Slides

X

20 Seconds



ivísimo





Process



Templates



Software

THINK



STAND BACK



**I'M GOING TO TRY
SCIENCE**

Agenda

- Pecha Kucha Talks
 - Gail Harris
 - Simon Brown
 - Will Chaparro
 - Ari Font
 - George Fairbanks
 - Eric Willeki
- What's in YOUR Silver Toolbox?

FIN.

Up Next...

Gail Harris
TVOntario

My Silver Toolbox Cache, Cache, Cache

tvo

Never stop learning

*"The more that you read, the more things you will know.
The more that you learn, the more places you'll go."*

—Dr. Seuss, I Can Read With My Eyes Shut!



- External caches
- Framework (embedded) caches
- Custom cache code

- External caches
- Framework (embedded) caches
- Custom cache code

External caches

- Varnish
- Memcache
- Alternative PHP Cache (APC)
- Content Delivery Network (CDN)

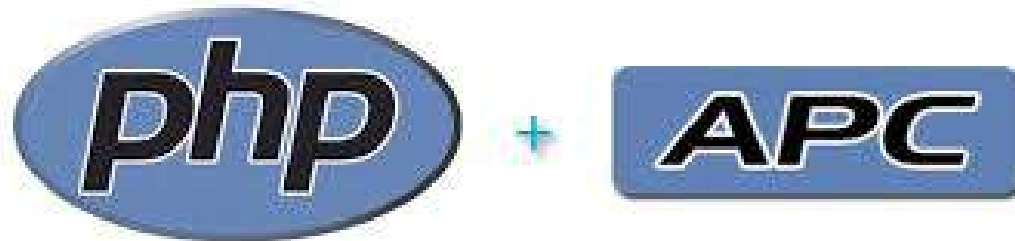
- **Varnish**
- Memcache
- Alternative PHP Cache (APC)
- Content Delivery Network (CDN)



- Varnish
- **Memcache**
- Alternative PHP Cache (APC)
- Content Delivery Network (CDN)



- Varnish
- Memcache
- **Alternative PHP Cache (APC)**
- Content Delivery Network (CDN)



- Drupal
 - page and block caching
- MySQL
 - query caching
 - key caching
- Apache
 - memory caching
 - file caching

- **Drupal**
 - page and block caching
- MySQL
 - query caching
 - key caching
- Apache
 - memory caching
 - file caching



- Drupal
 - page and block caching
- **MySQL**
 - query caching
 - key caching
- Apache
 - memory caching
 - file caching



- Drupal
 - page and block caching
- MySQL
 - query caching
 - key caching
- **Apache**
 - memory caching
 - file caching



- Story 1
- Story 2
- Story 3

- **Story 1**
- Story 2
- Story 3

- Story 1
- **Story 2**
- Story 3

- Story 1
- Story 2
- **Story 3**

- Measure first, target your efforts
- Three Strategies:
 - External caches
 - Framework (embedded) caches
 - Custom cache code
- Performance tuning takes time and practice



Ignite your potential.

tvo Never stop learning



Feed your mind.

tvo Never stop learning



What happens when you open your mind?

tvo Never stop learning

Are your shoelaces tied?
Everyone check now.

tvo
tvo

Never stop learning

FIN.

Up Next...

Simon Brown
Coding the Architecture
@simonbrown

Diagram-driven design

(with a hat-tip to Gregor Hohpe
for the title of this talk)

Structure

Understand the significant structural elements and how they fit together, based upon the architectural drivers.

Design and decomposition down to containers and components.

Vision

Create and communicate a vision for the team to work with.

Context, container and component diagrams.

Risks

Identify and mitigate the highest priority risks.

Risk-storming and concrete experiments.

Just enough up front design
to create firm foundations
for the software product and its delivery



Cezar Saramet

@cezarscv



Follow

You can see the ability to think in an abstract way as the ability to not get caught up in the details all of the time. [@simonbrown](#)



RETWEETS

4

FAVORITE

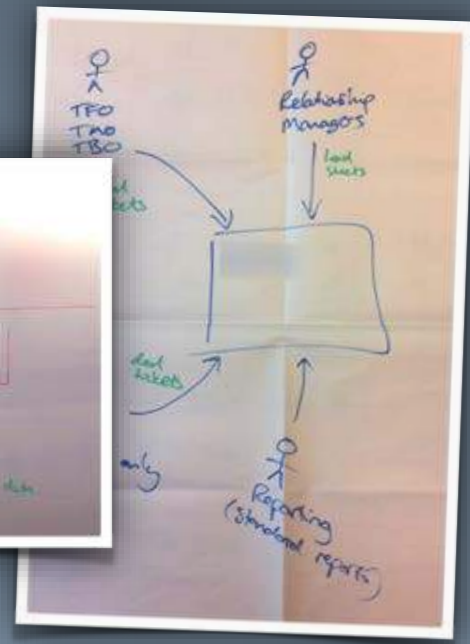
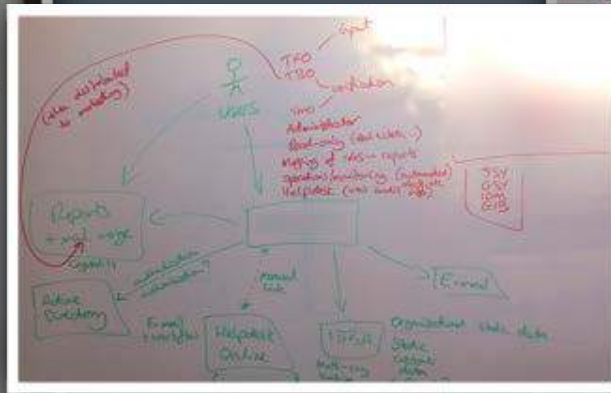
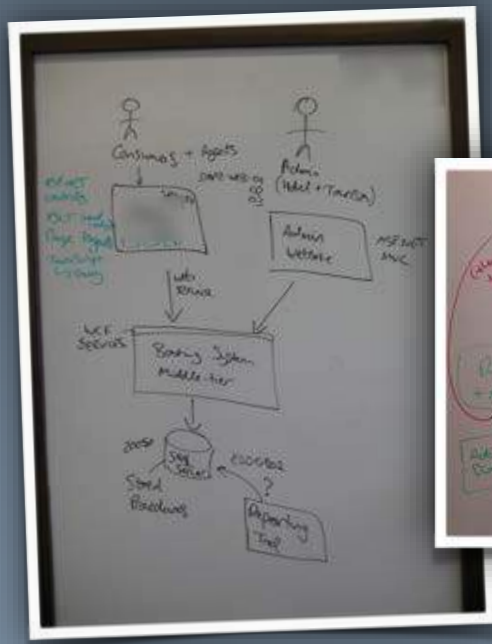
1

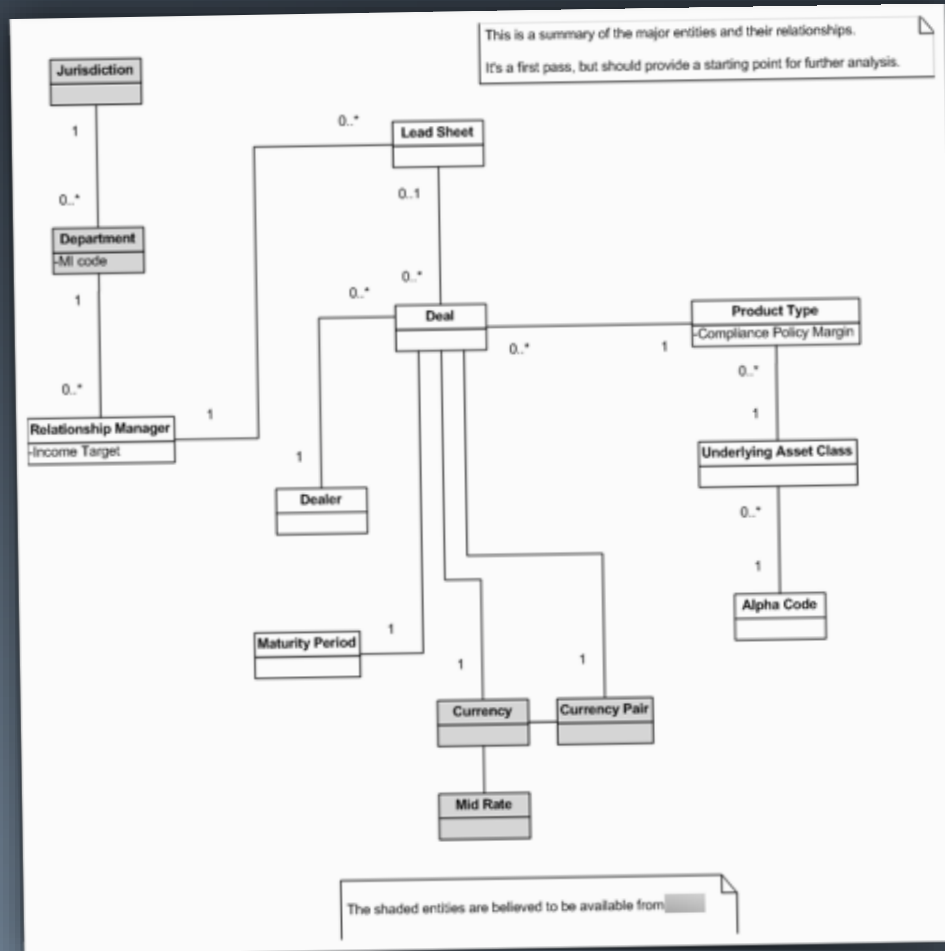


2:10 PM - 19 Apr 2015

Whiteboards

and context diagrams

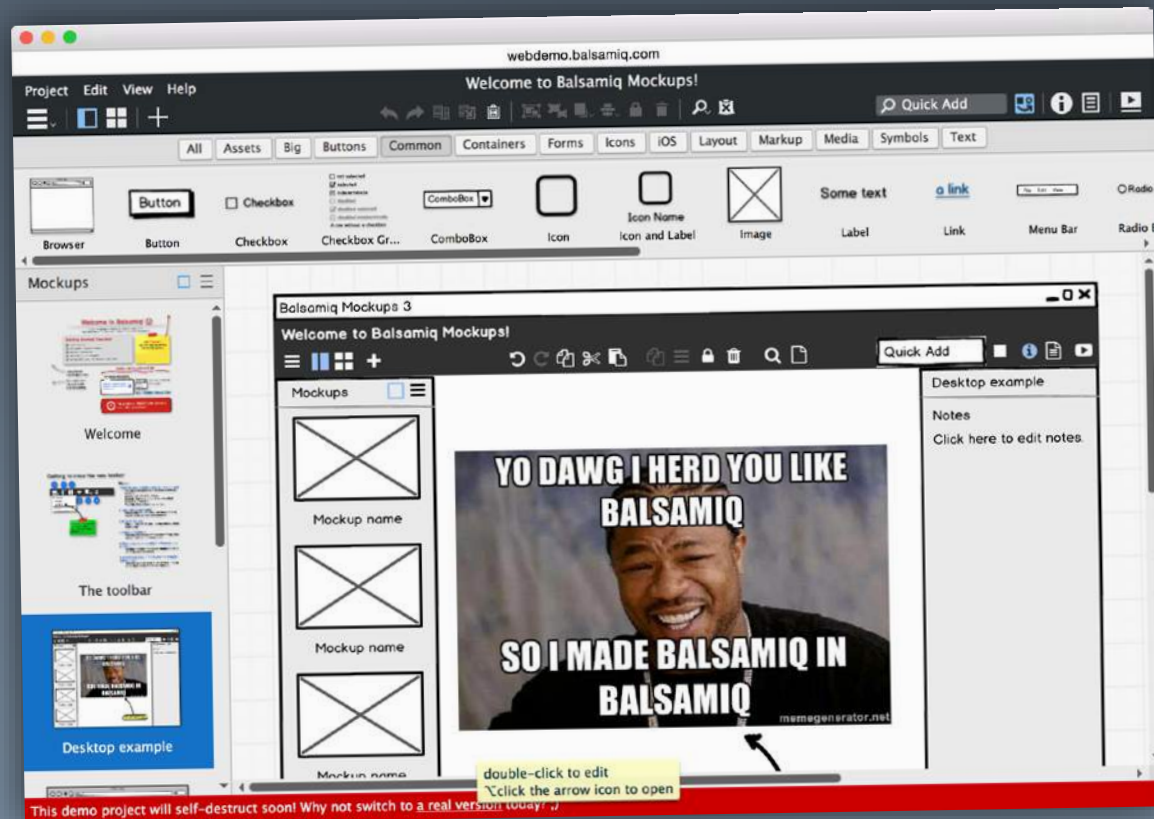




Domain models

Wireframes

(e.g. Balsamiq)



Components

~~Classes,~~

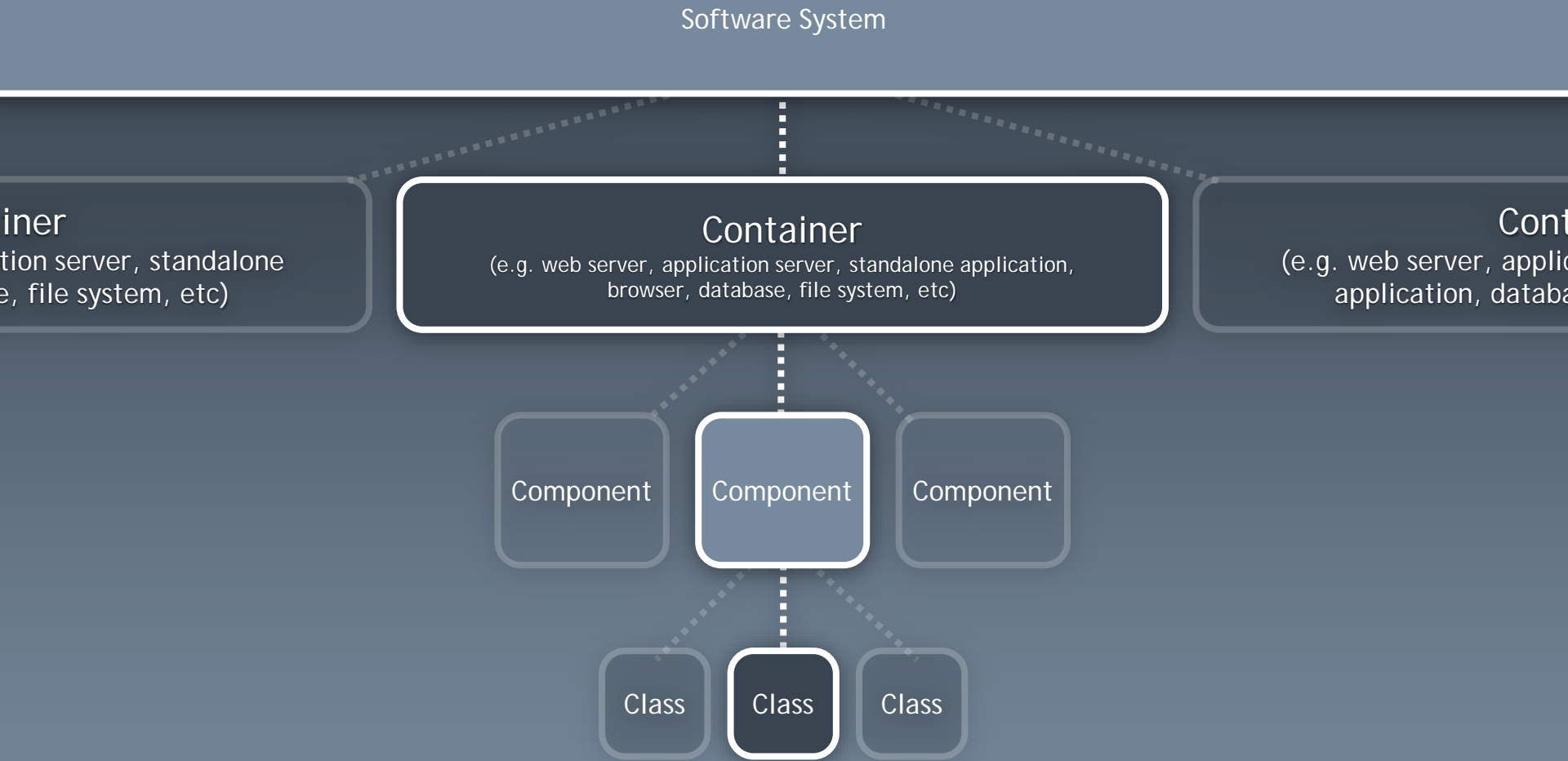
Responsibilities,
Collaborations

[illegible]

* two or more people

A common set of
abstractions

is more important than
a common notation



Agree on a simple set of **abstractions** that
the whole team can use to communicate

The C4 model



System Context

The system plus users and system dependencies



Containers

The overall shape of the architecture and technology choices



Components

Logical components and their interactions within a container

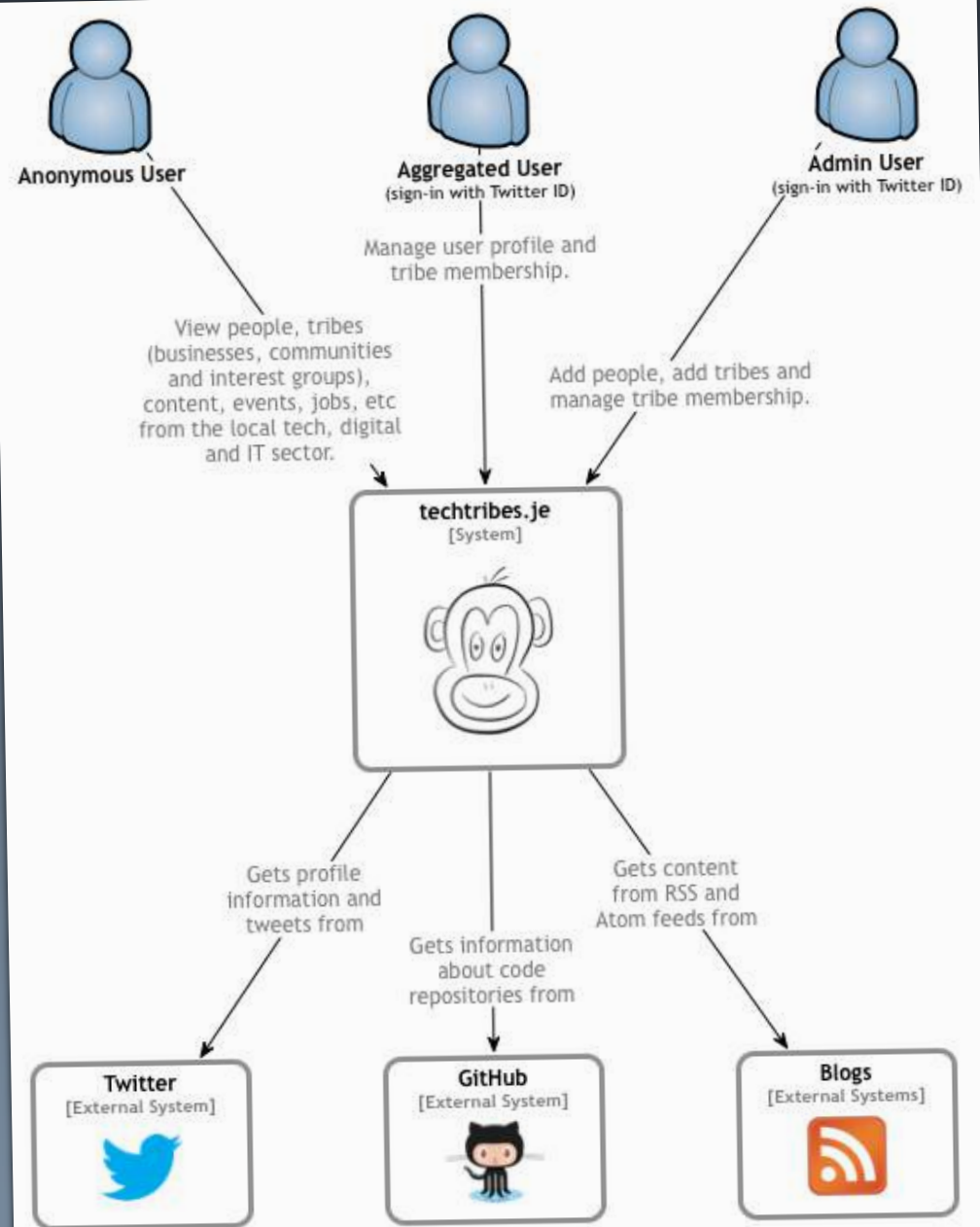


Classes

Component or pattern implementation details

Context

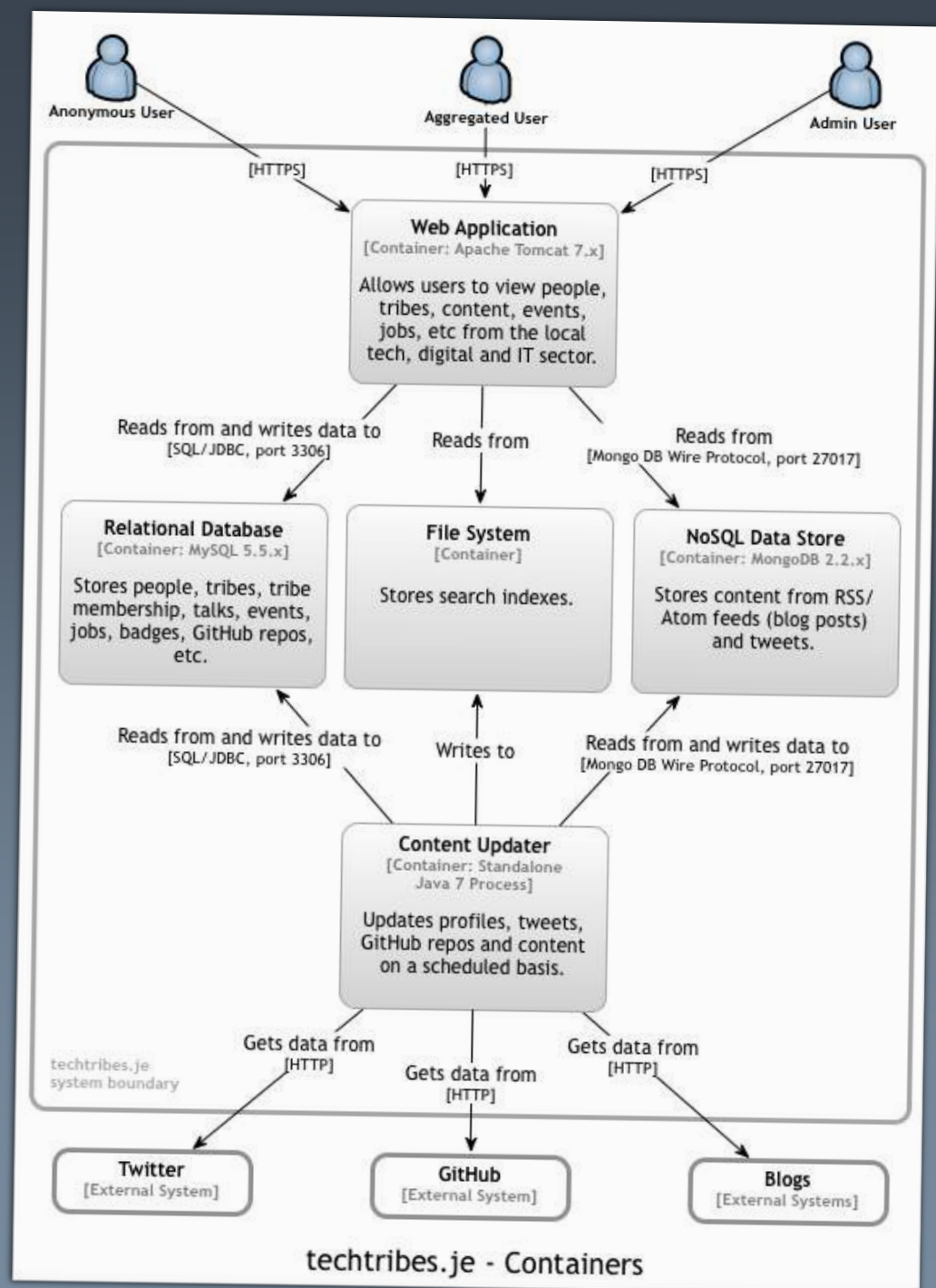
- What are we building?
- Who is using it?
(users, actors, roles, personas, etc)
- How does it fit into the existing IT environment?
(systems, services, etc)



techtribes.je - Context

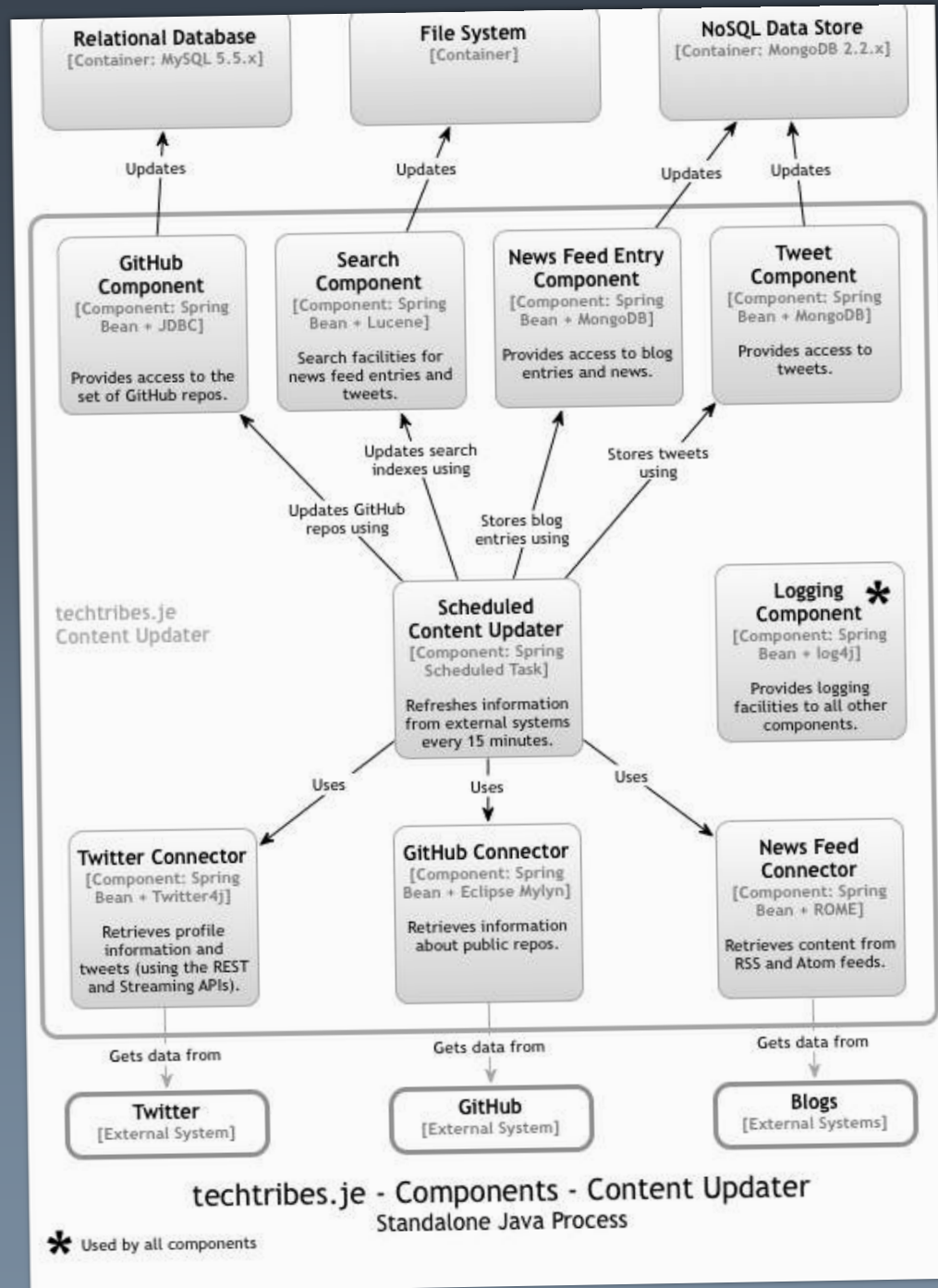
Containers

- What are the high-level technology decisions? (including responsibilities)
- How do containers communicate with one another?
- As a developer, where do I need to write code?



Components

- What components/services is the container made up of?
- Are the technology choices and responsibilities clear?





Diagrams are maps
that help a team navigate a complex codebase

A notationless notation
(whiteboard and sticky note friendly,
supplemented with colour coding)

My Web Application

[Container: Apache Tomcat 7.x]

Here is a list of the key
responsibilities for my
web application.

Titles

Short and meaningful, numbered if diagram order is important

Lines

Make line style and arrows explicit, add annotations to lines to provide additional information

Layout

Sticky notes and index cards make a great substitute for drawn boxes, especially early on

Labels

Be wary of using acronyms

Colour

Ensure that colour coding is made explicit

Orientation

Users at the top and database at the bottom? Or perhaps “upside-down”?

Shapes

Don't assume that people will understand what different shapes are being used for

Keys

Explain shapes, lines, colours, borders, acronyms, etc

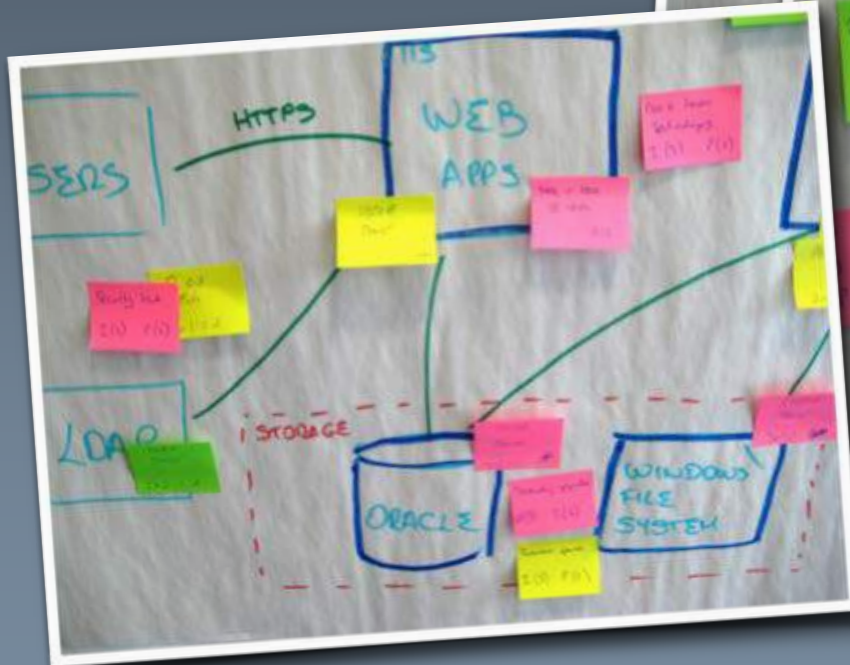
Responsibilities

Adding responsibilities to boxes can provide a nice “at a glance” view (Miller's Law; 7 ± 2)

Some tips for
effective sketches



Risk-storming



A collaborative and visual technique for identifying risk

Structurizr

No more messing with drawing tools. Create software architecture models and diagrams as code based upon the [C4 software architecture model](#).

Simple

No more endless hours spent messing with drawing tools figuring out what to include and then manually drawing boxes and lines.

Versionable

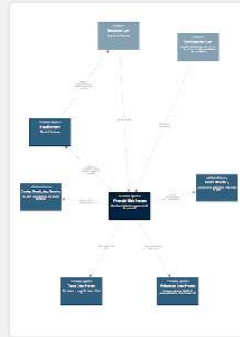
The software architecture model can be easily versioned along with your code.

Up-to-date

Integration with your build process means your software architecture model can be continuously kept up to date.

Scalable

Static diagrams are hard to change and work with once they start to get large. Having the software architecture model as code puts you in control.



Architecturally-evident
coding styles and
software architecture
models as code

From code

You create a software architecture model by extracting the

```
Workspace work  
Model model =
```

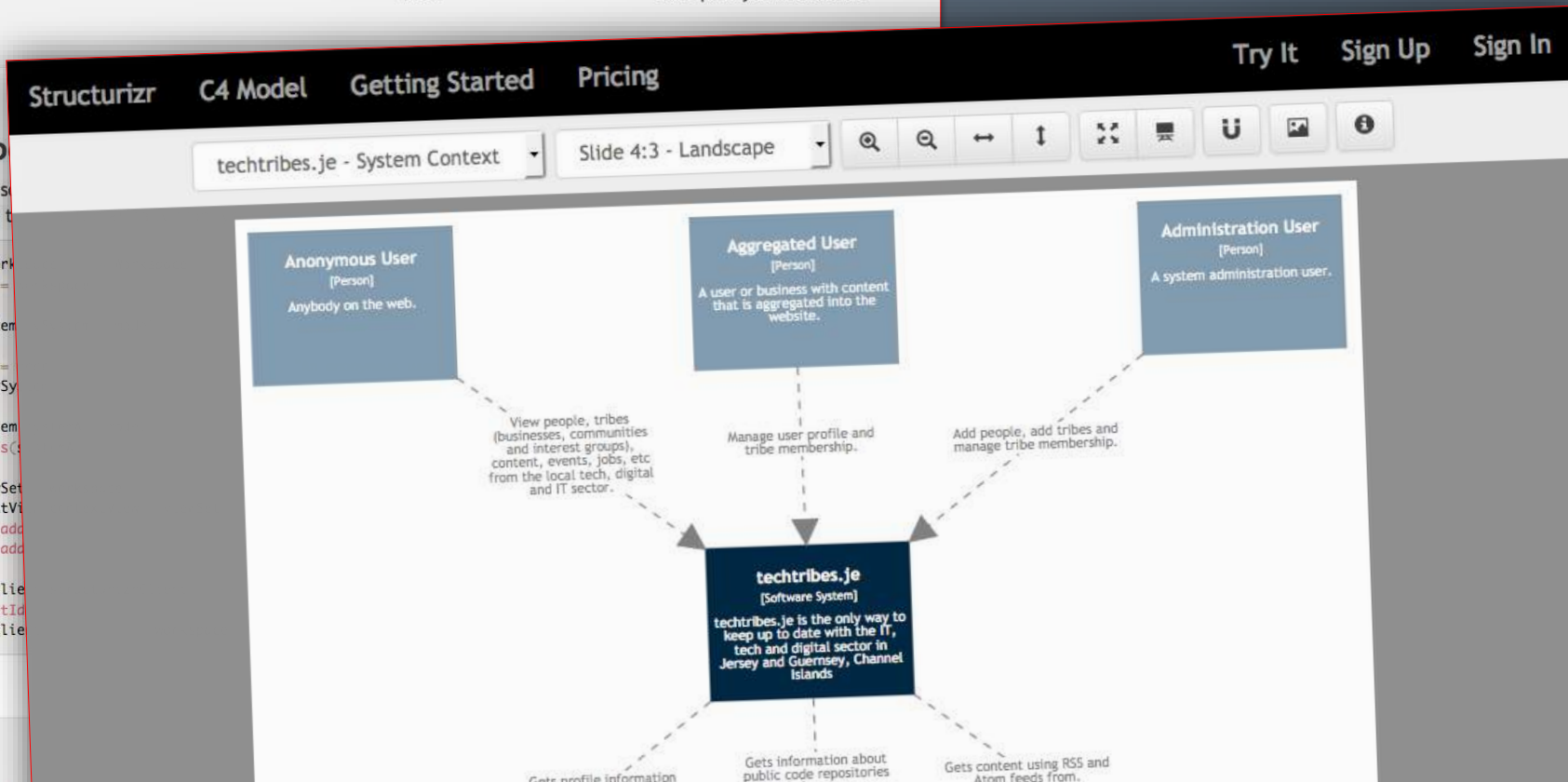
```
SoftwareSystem
```

```
Person user =  
user.uses(mySys
```

```
SoftwareSystem  
mySystem.uses(s
```

```
ViewSet viewSet  
SystemContextVi  
contextView.add  
contextView.add
```

```
StructurizrClient  
workspace.setId  
structurizrClient
```



Aligning software architecture and code

Sketches

for early and quick up front design

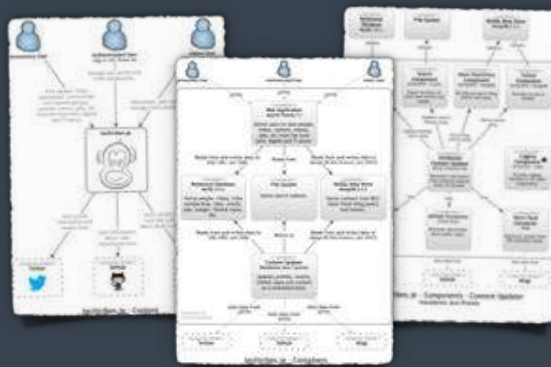
for increased modularity plus easier inspect and adapt loops

Software architecture as code

for models that are continuously kept up to date

Risk-storming

for identifying your highest priority risks



The C4 model and simple software architecture diagrams

A ubiquitous language

for good communication and moving fast

Agility and the essence of software architecture
(creating agile software systems in an agile way)

FIN.

Up Next...

Will Chaparro
IBM
@wmchaparro

Project Inception Deck

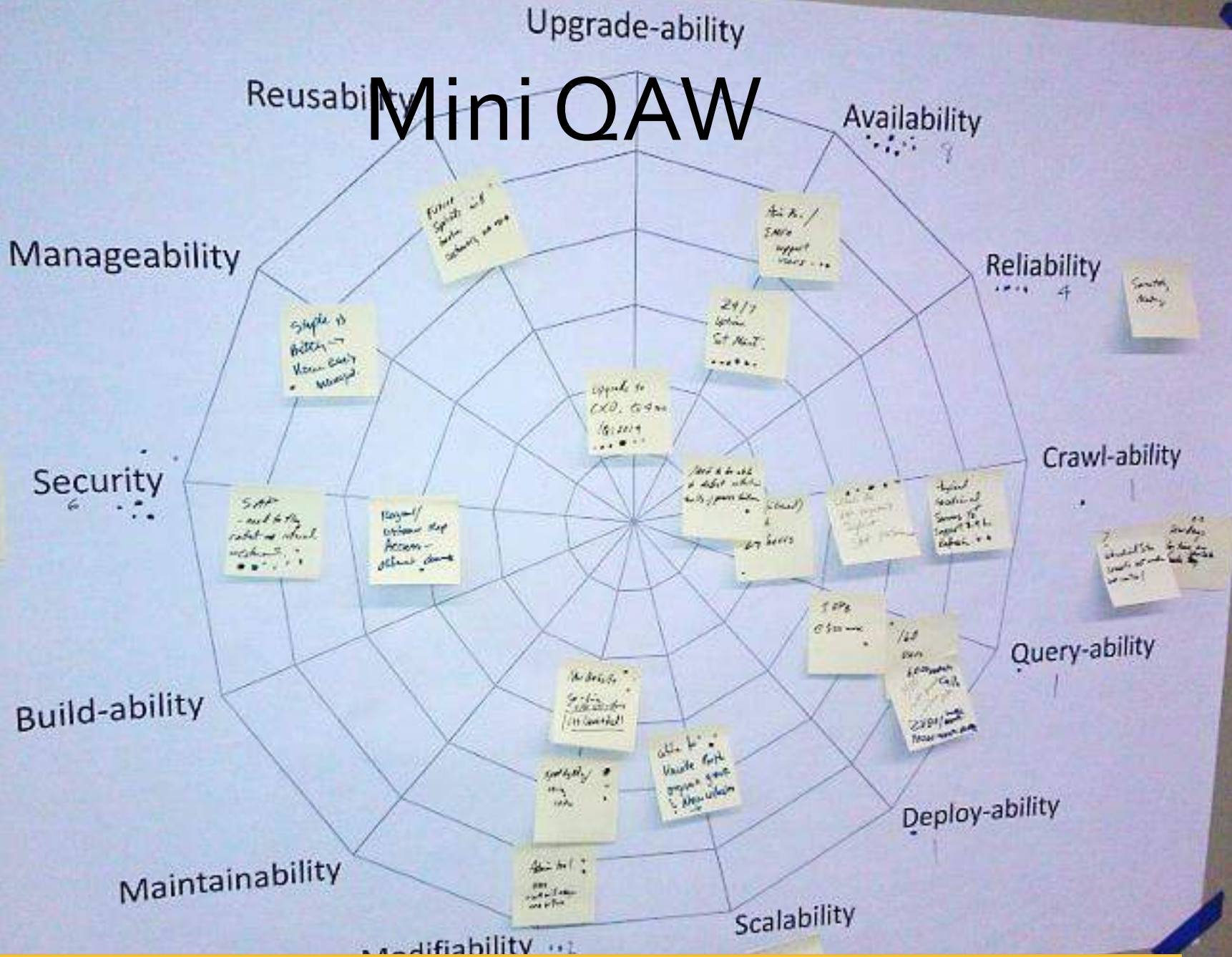
- Project “Charter”
 - Why are we here?
 - Elevator Pitch
 - Business Value
 - Scope Concurrence
 - Stakeholders
 - Notional Architecture
 - Risks
 - Timeline
 - Tradeoff Sliders
 - Team
- 

Why its Effective

- Alignment
- Expectation Setting



Mini QAW



Why its Effective

- Fast
- Repeatable
- Relatable
- Trainable
- Reliable

Grow don't Build



Why its Effective

- Build a system that can grow and evolve over time
- Requirements change over time
- Allows you to identify problems easily and early



Fail fast, learn often



Why its Effective

- Learn, adjust your course more quickly
- Its OK to take risks and innovate



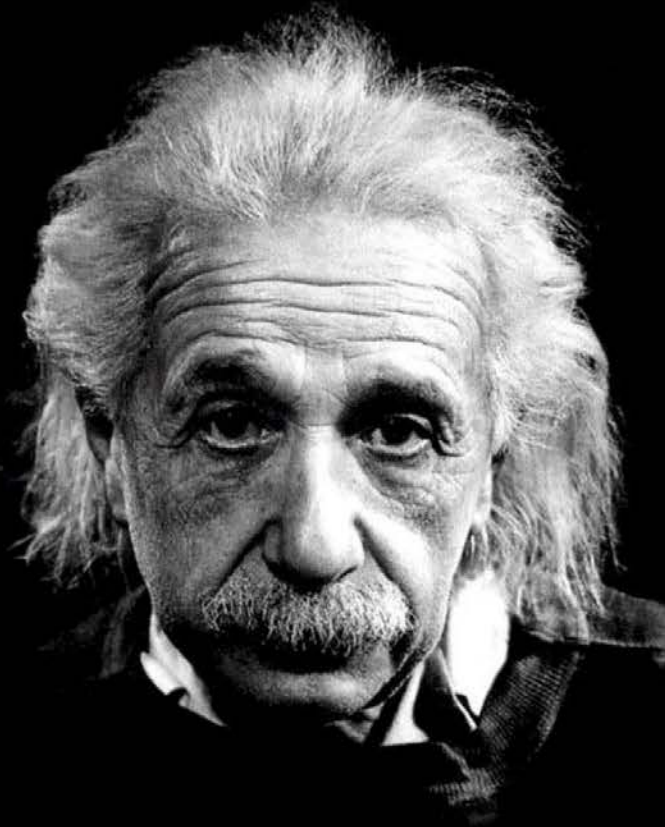
Rate the Meeting



Why its Effective

- Most meetings fail
- You immediately know if the meeting didn't fail or if it did
- You can start analyzing why your meetings still fail and DO SOMETHING ABOUT IT!

Bring in the Expert



Sometimes you just need to bring in the expert

Why its Effective



Listen



Why its Effective

- The speaker feels important
- The speaker feels confident in expressing their feelings
- The speaker feels understood

Don't take things personally



Why its Effective

- We are passionate about our own ideas
- Our own ideas aren't always right
- But if they are, you need to convince others





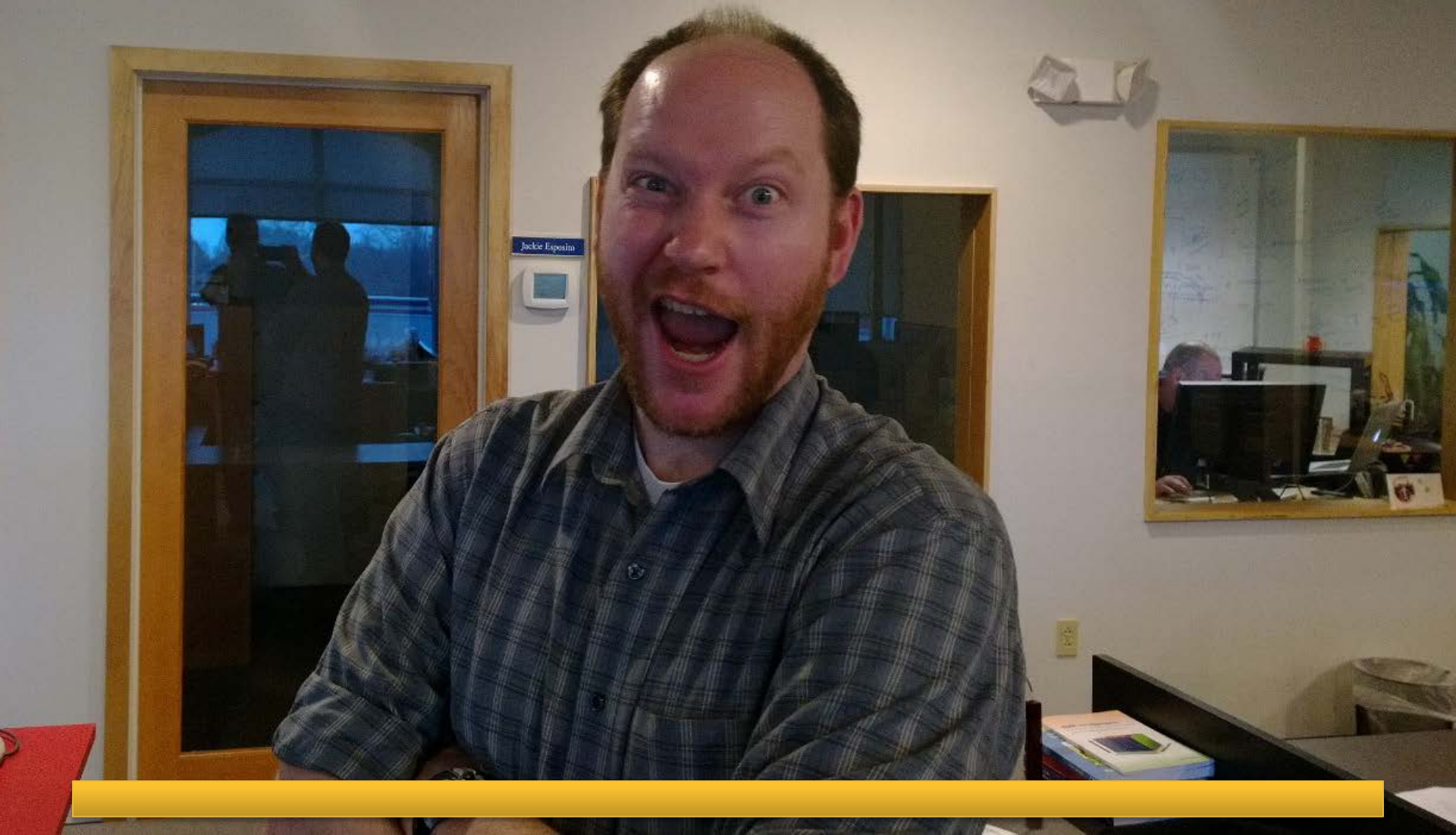
Have Fun

Why its Effective

We are more
productive when we
are happy



That guy (or gal)



Why its Effective

- Its nice to bounce ideas off of someone
- Feedback loops
- Better communication = better working software

FIN.

Up Next...

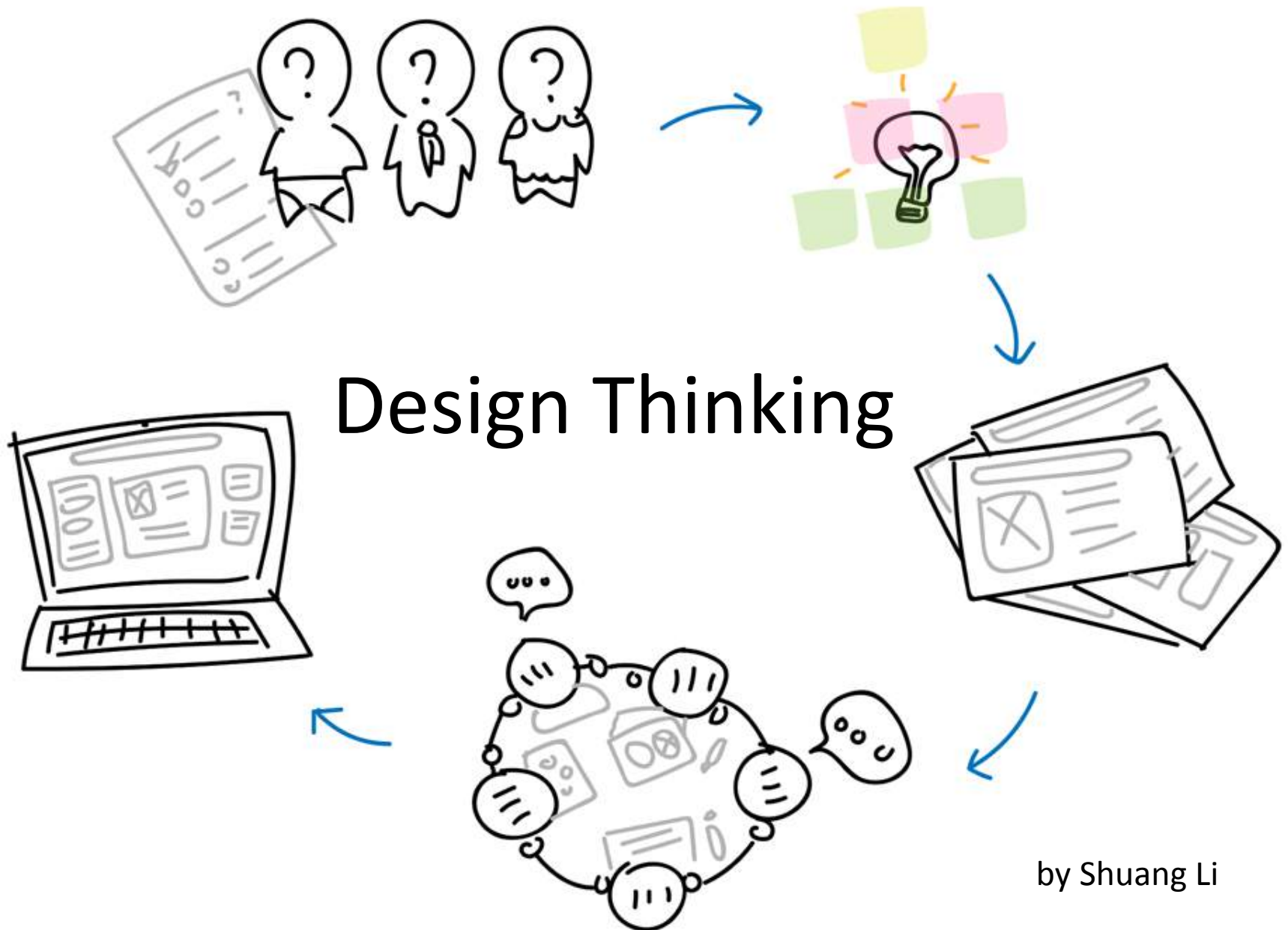
Ariadna Font
IBM
@quicola

Deliver user value
incrementally

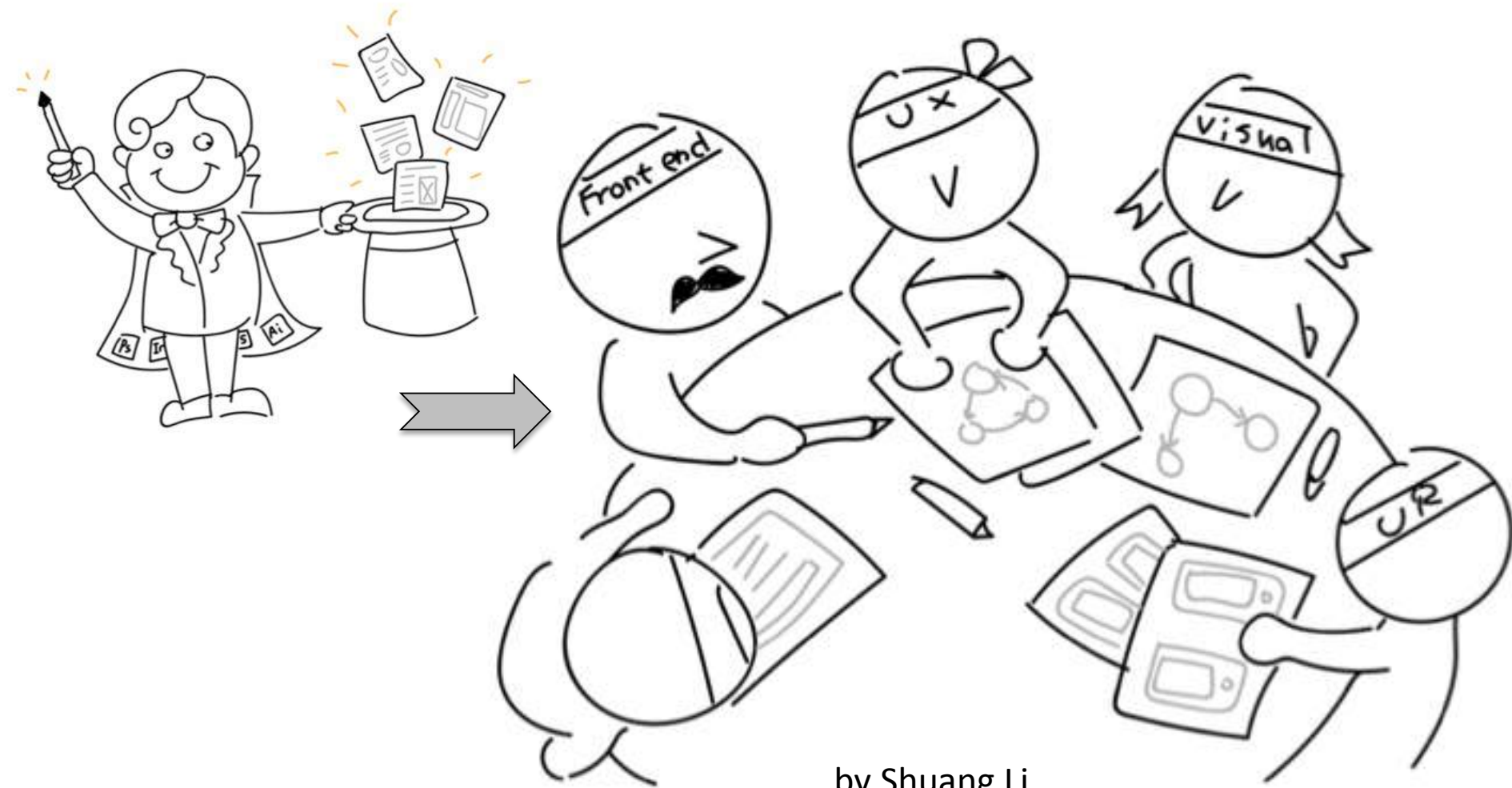
(Ari's silver toolbox)

SATURN 2015

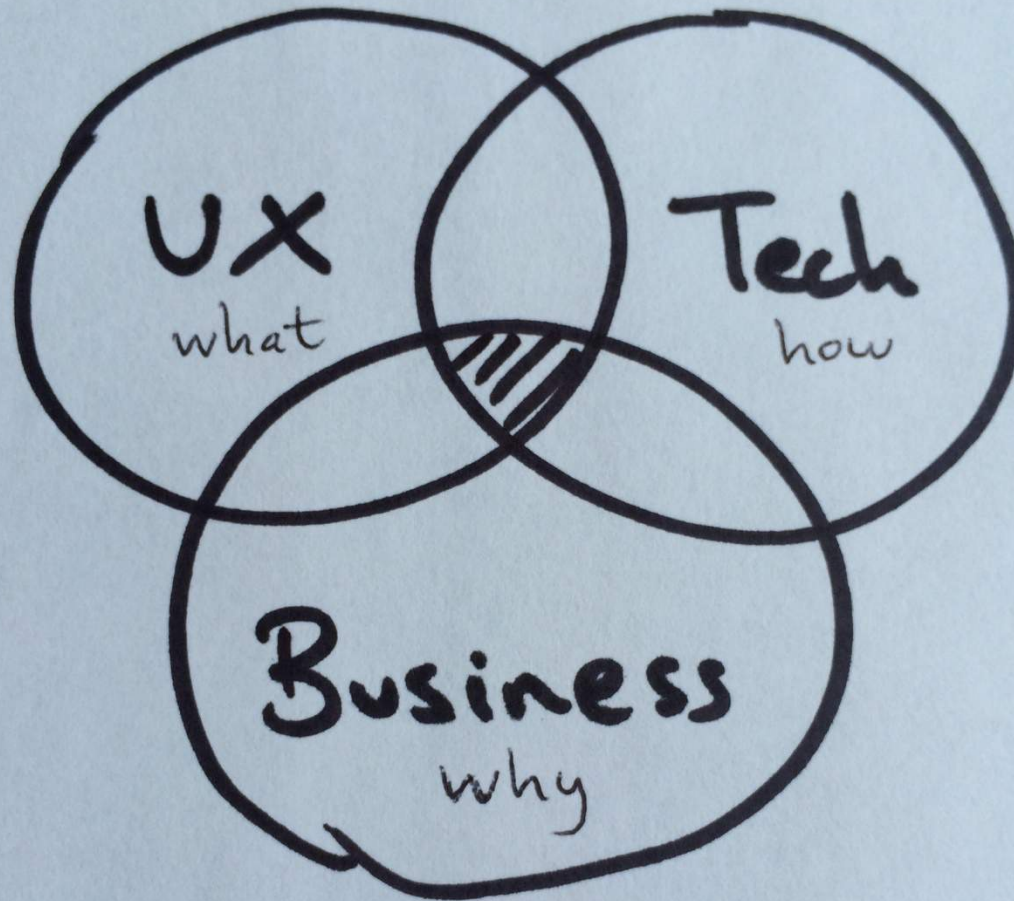
Pecha Kucha



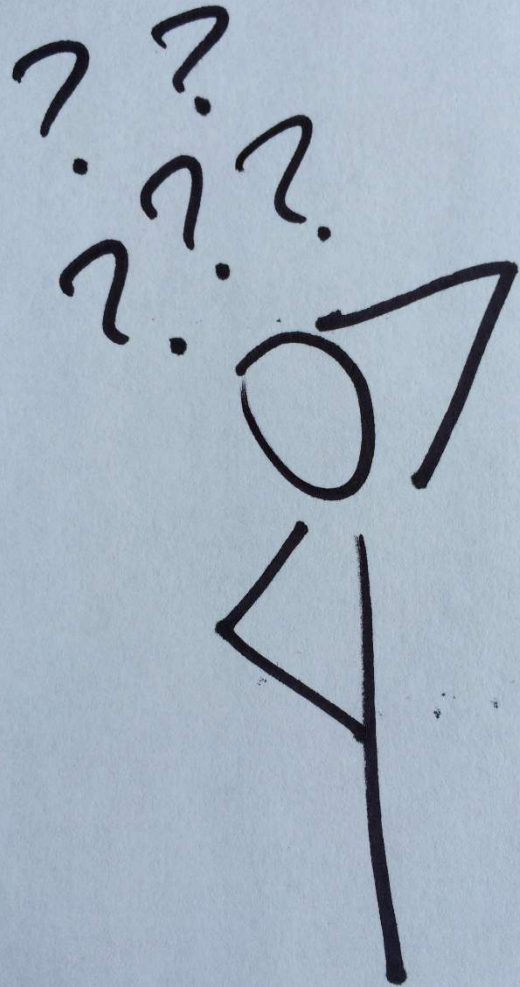
Old way vs New way



by Shuang Li

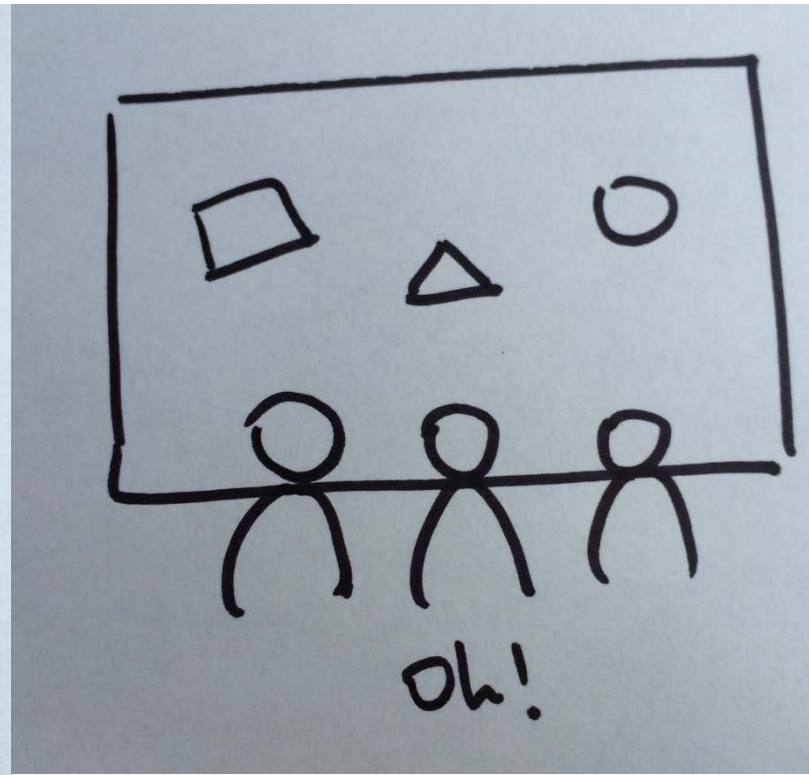
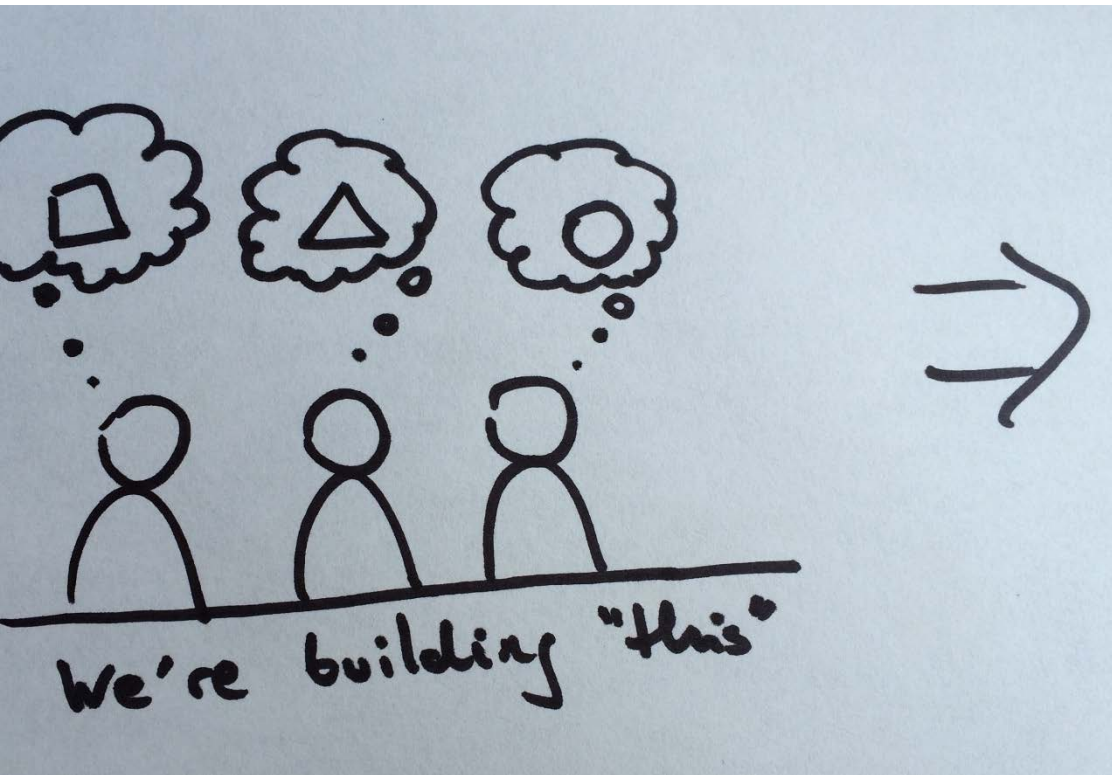


Always start with the why

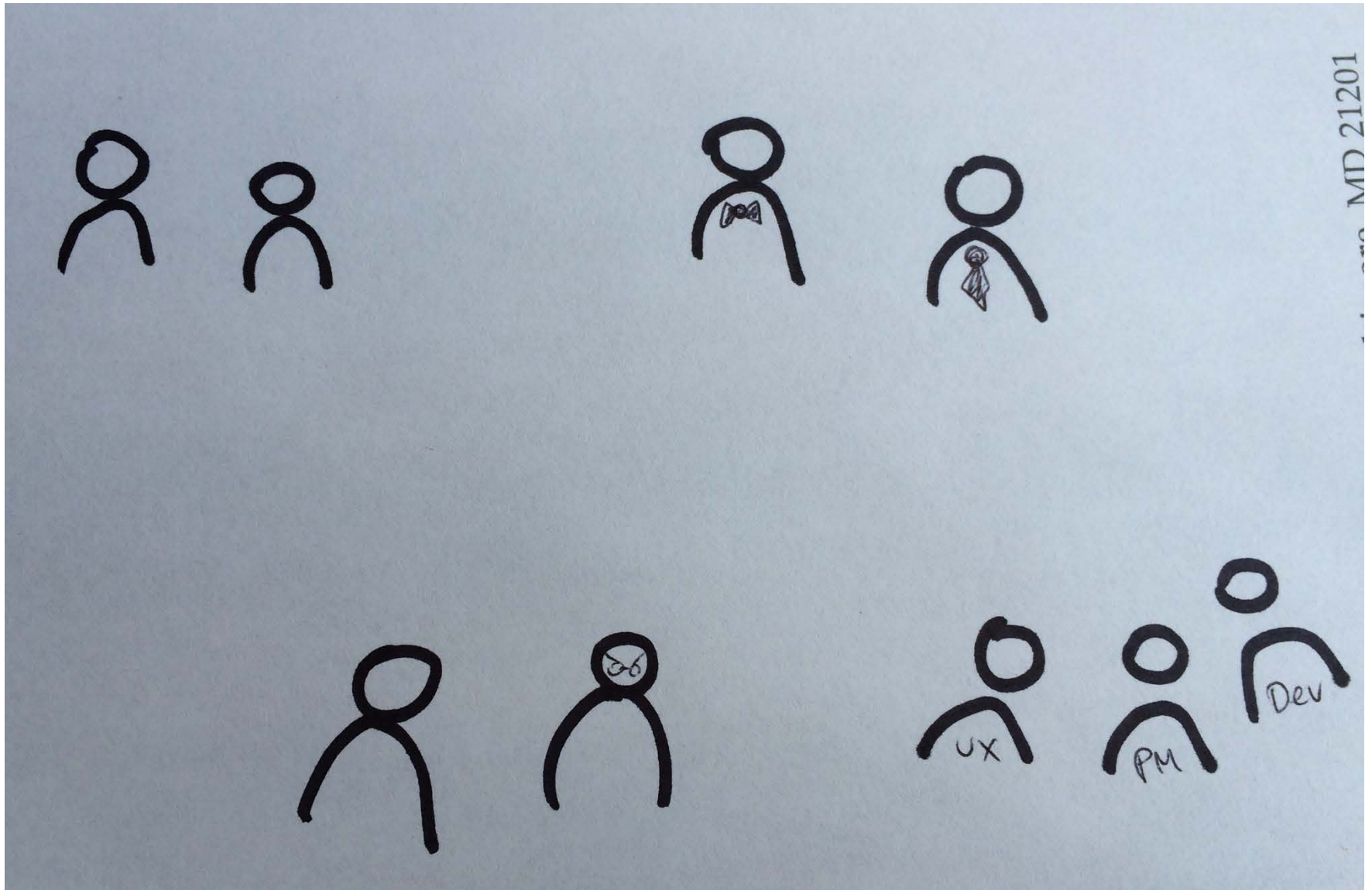


? \Rightarrow hypothesis 1
?
? \Rightarrow hypothesis 2
?
? \Rightarrow hypothesis 3

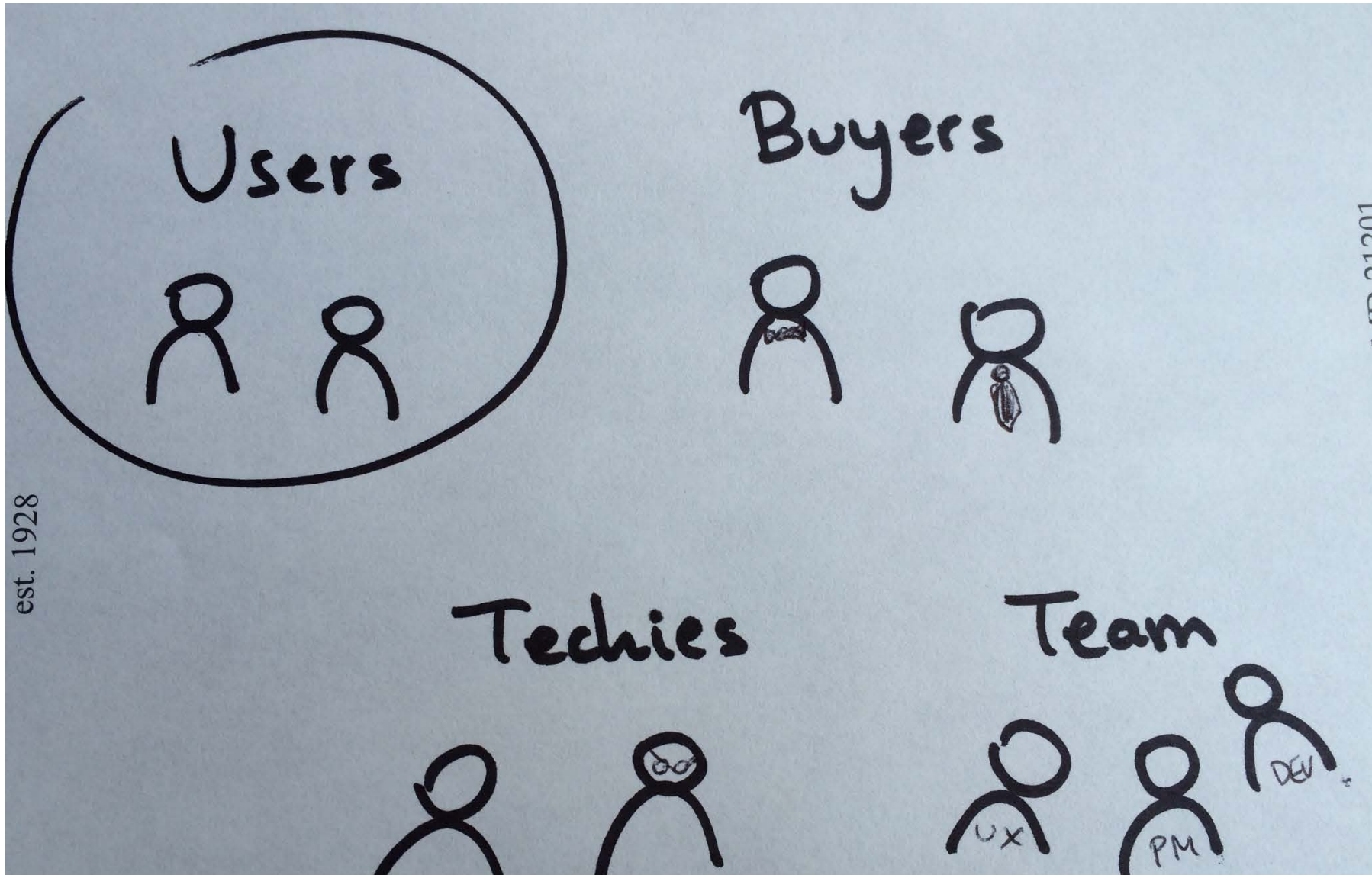
Facilitate shared understanding



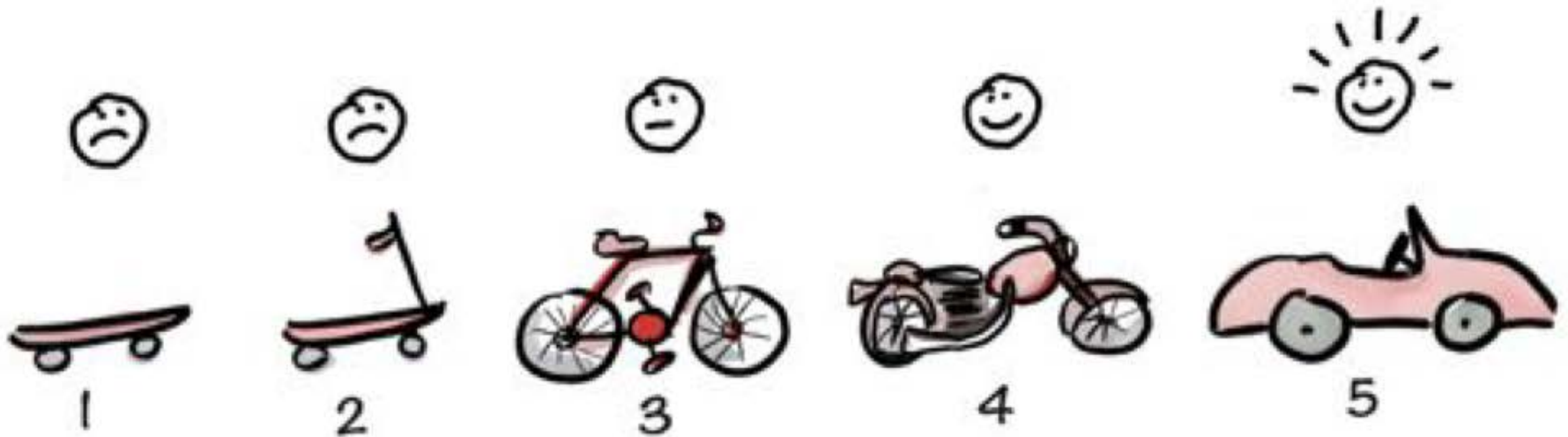
Stakeholder map



Who is the user?



Incrementally delivering user value



by Henrik Kniberg



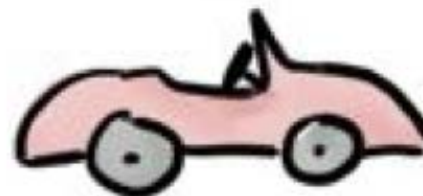
1



2



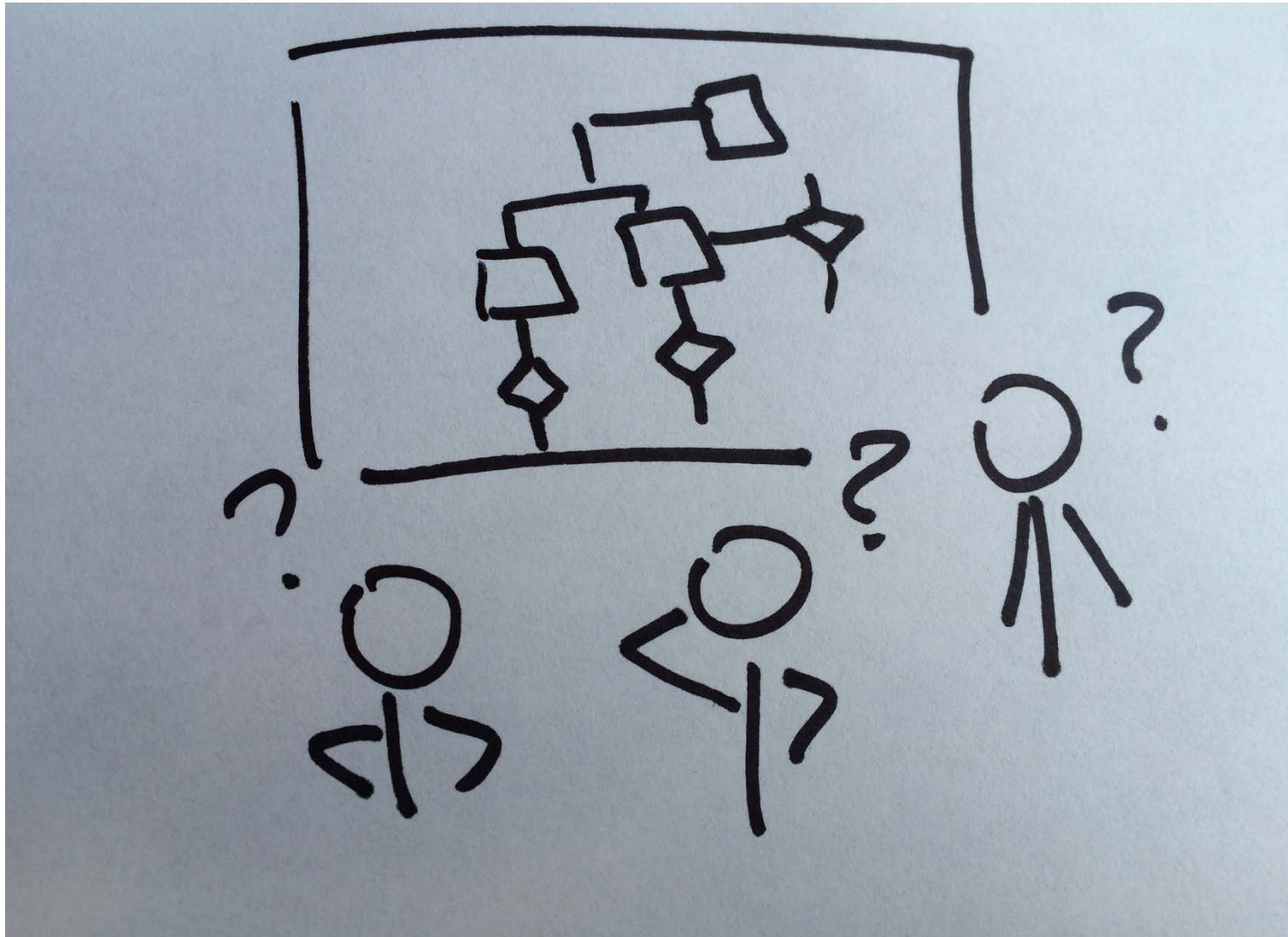
3



4

by Henrik Kniberg

Who is the user of the architecture?



How can we learn more about our users?

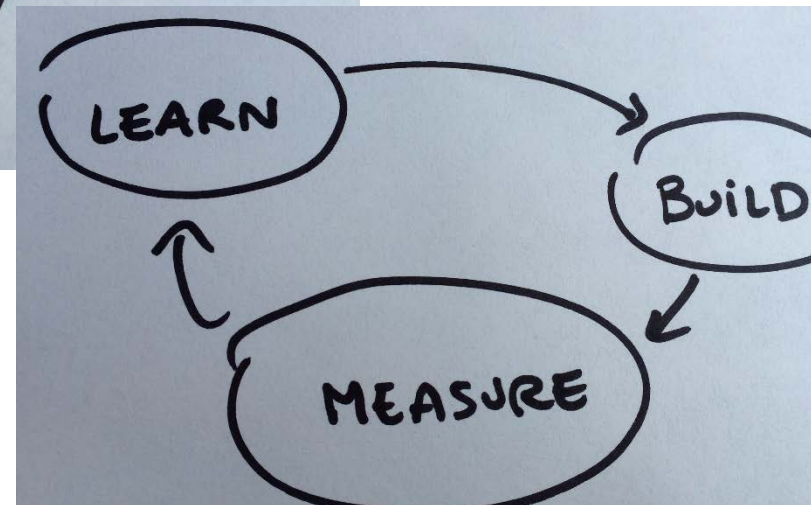
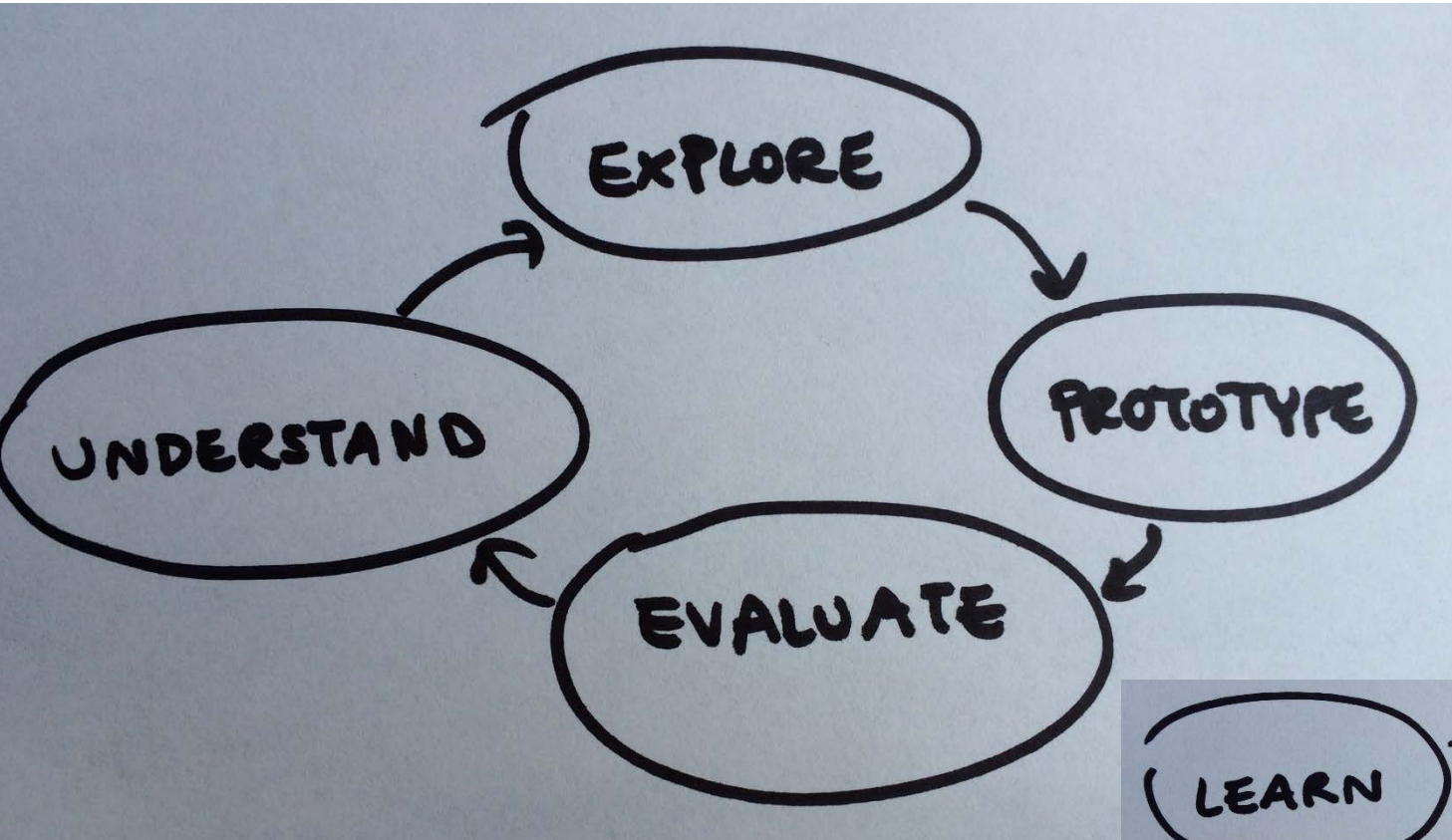
- Go where the users are
- Have conversations
- Talk to them about their work
- Watch them work

Easier with internal stakeholders

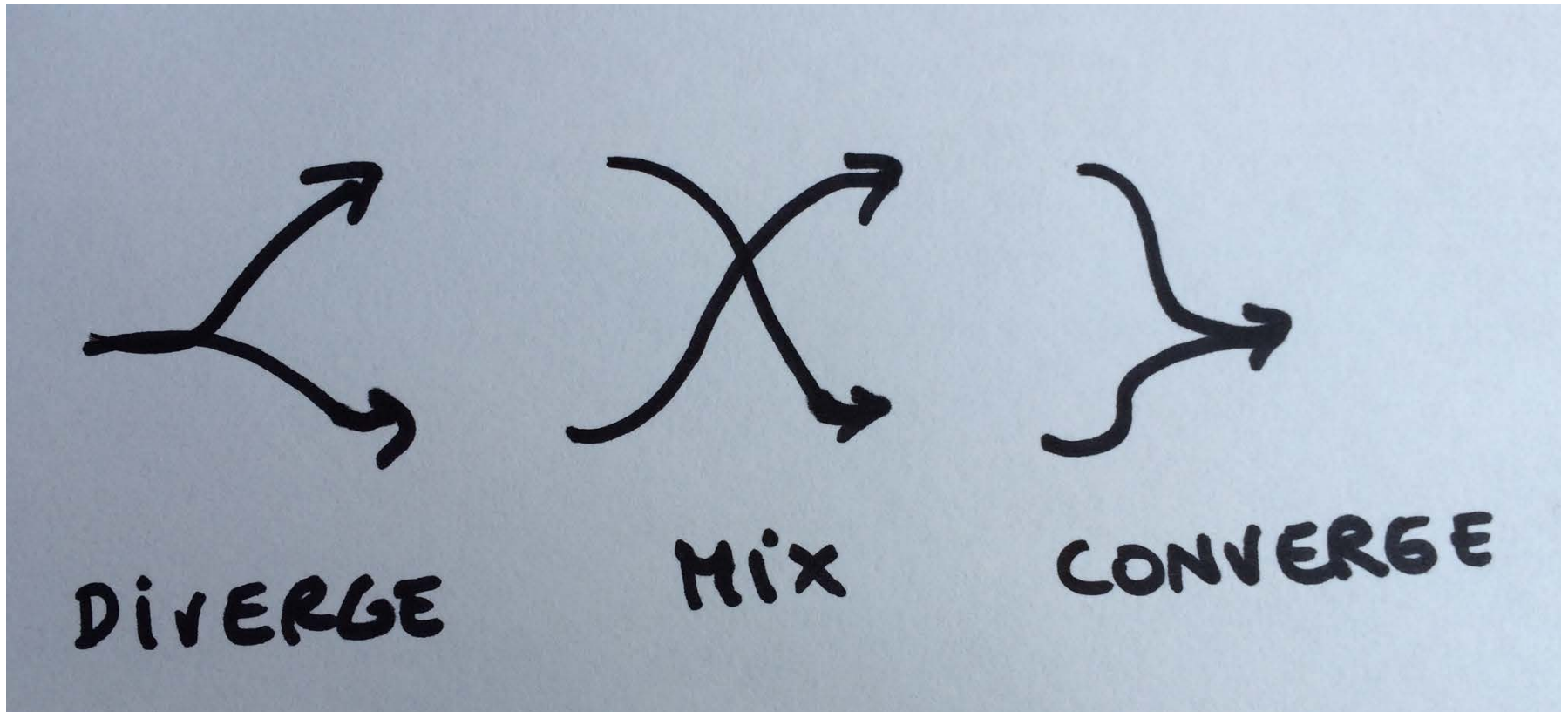
What other constraints we should consider now (vs later)

- Technical feasibility
(architecture, back-end, front-end, UI, ...)
- Timeframe
- Resources
- Team size
- Team skills and domain expertise
- ...

Find a starting point and iterate



When you can't solve it by yourself...



When in doubt...

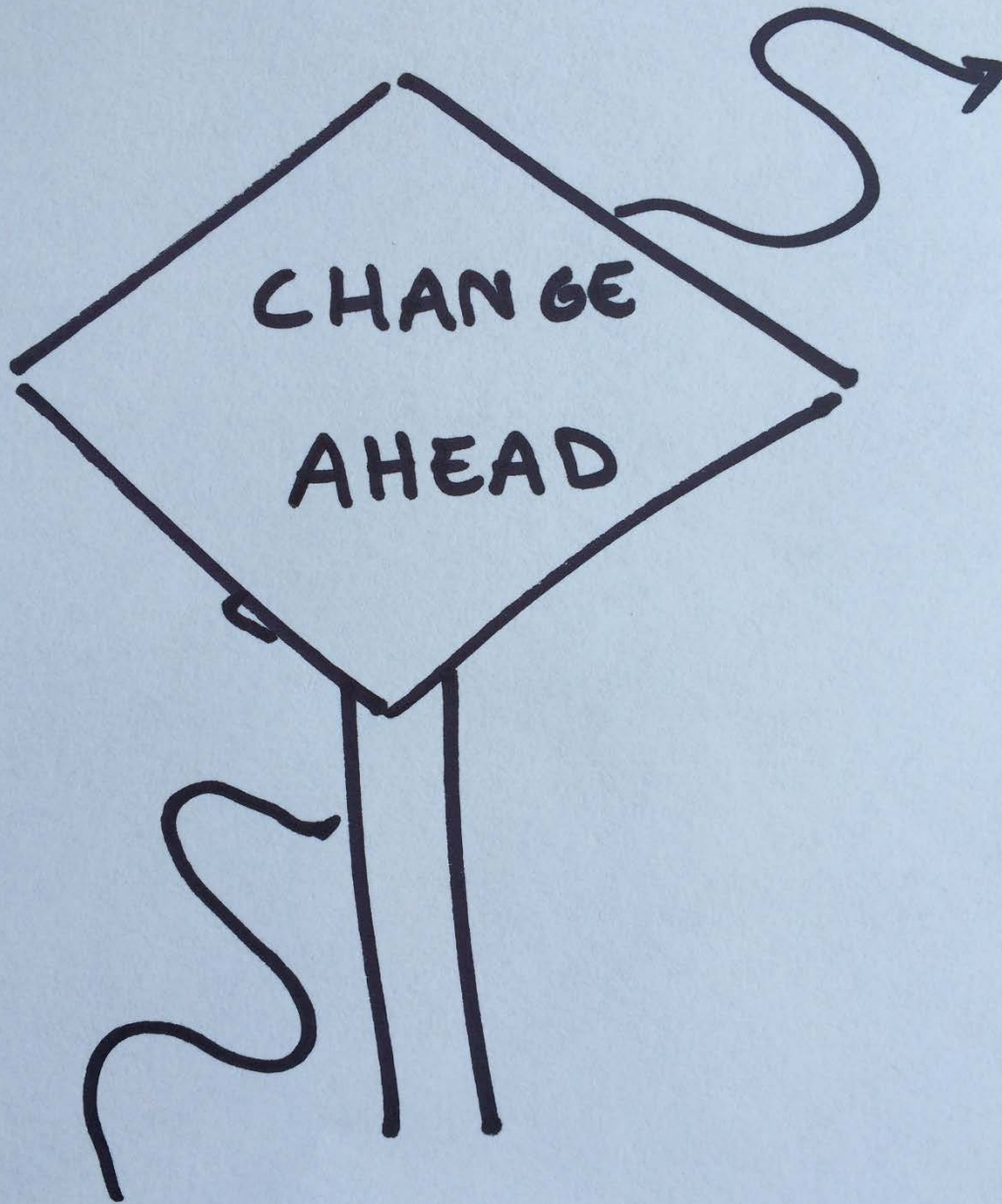
- Playback what you have learned to the rest of the team and find out what they have learned

When in doubt...

- What is the new set of assumptions and hypotheses that you need to validate to make progress?

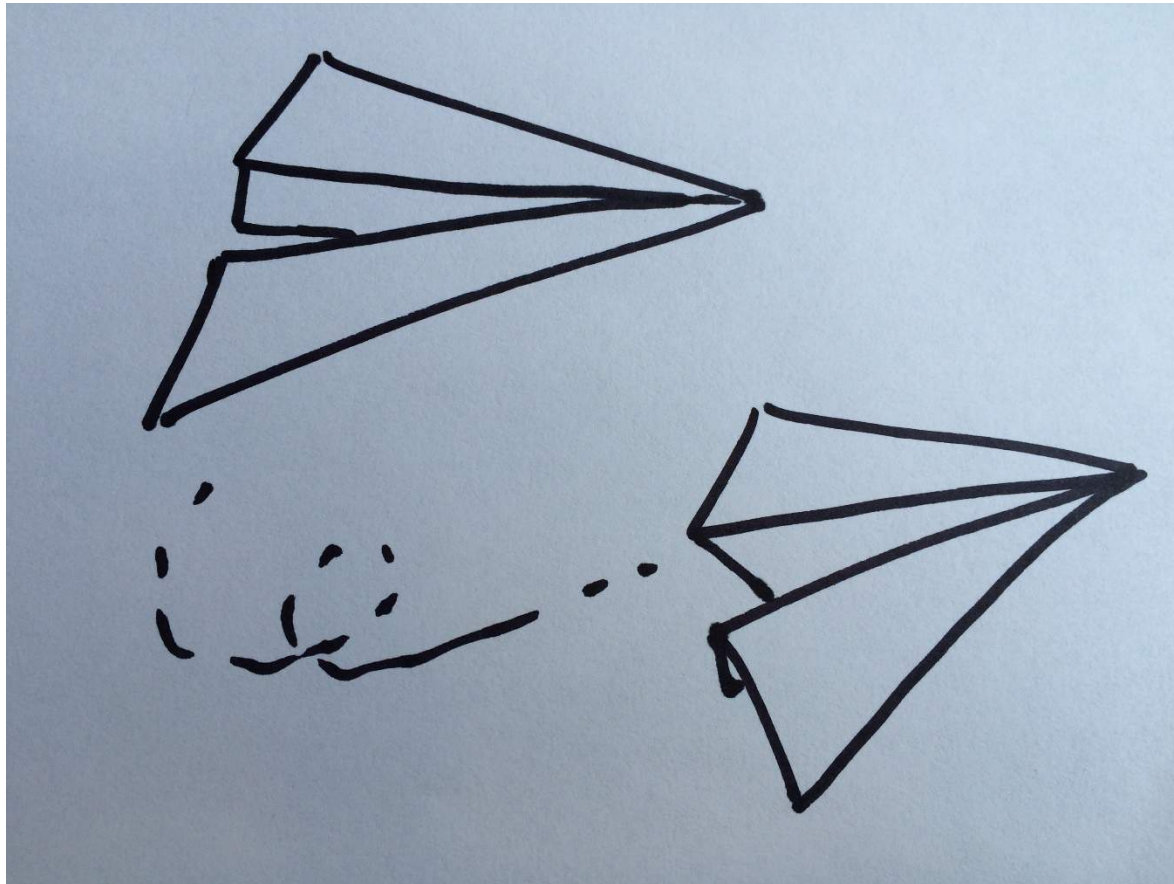
When in doubt...

- Go test them
 - talk to key stakeholders
 - build a prototype
 - dance
 - whatever you need to do



In a complex
domain,
optimize for
ease of change
over ease of
predictability

What's the cheapest and fastest way
to (in)validate your architecture,
model?



FIN.

Up Next...

George Fairbanks
Google
@GHFairbanks



My Silver Toolbox: Building Models Quickly and Carefully

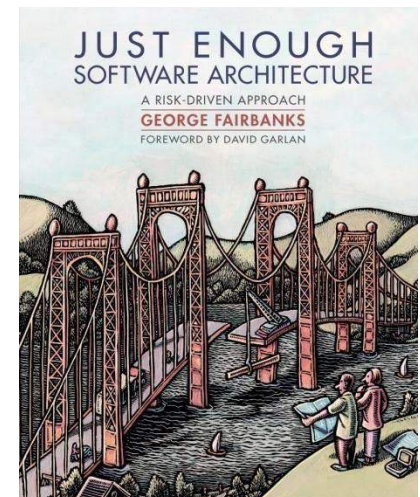
George Fairbanks

SATURN 2015
30 April 2015

Rhino Research

<http://RhinoResearch.com>

<http://GeorgeFairbanks.com>



Scenario: **Library checks out Moby Dick**

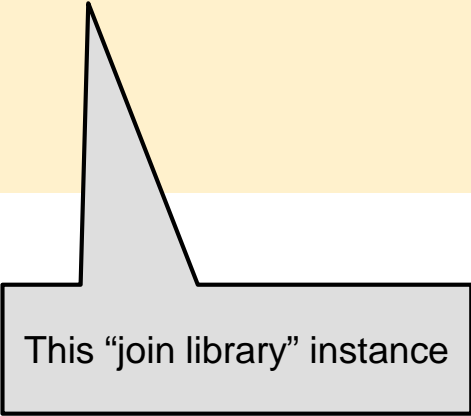
Steps:

1. Pat joins the NYC public library.

Scenario: **Library checks out Moby Dick**

Steps:

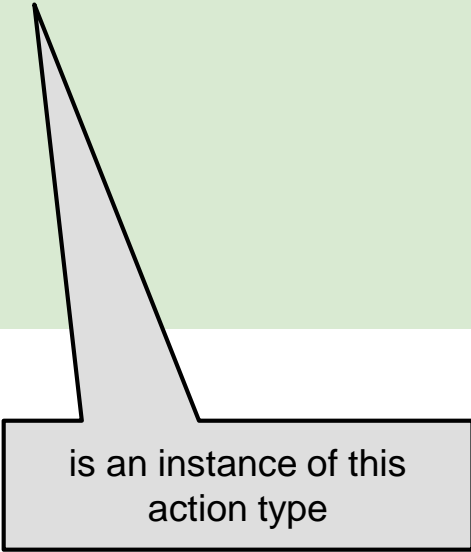
1. Pat joins the NYC public library.



This “join library” instance

Actions

- Join Library



is an instance of this
action type

Scenario: **Library checks out Moby Dick**

Steps:

1. Pat (Patron) joins the NYC public library.

Actions

- Join Library (Patron)

Instance

This patron example

Pat : Patron

corresponds to this
instance

Type

Patron

and is an instance of this
general type

Scenario: **Library checks out Moby Dick**

Steps:

1. Pat (Patron) joins the NYC (Library).

Actions

- Join Library (Patron)

Ditto for the library

Instance

NYC : Library

Pat : Patron

Pat is a member of this
specific library

Type

Library

*

*

Patron

but in general, could join
many libraries.

Scenario: **Library checks out Moby Dick**

Steps:

1. Pat (Patron) joins the NYC (Library).
2. Pat searches for titles about fish (Topic).

Next step in the scenario

Actions

- Join Library (Patron)
- Search titles (Topic)

corresponding action

Instance

NYC : Library

Pat : Patron

some specific topics

Nature : Topic

Fish : Topic

Type

Library

*

*

Patron

and the general idea

Topic

Scenario: **Library** checks out **Moby Dick**

Steps:

1. Pat (Patron) joins the NYC (Library).
2. Pat searches for titles about fish (Topic).

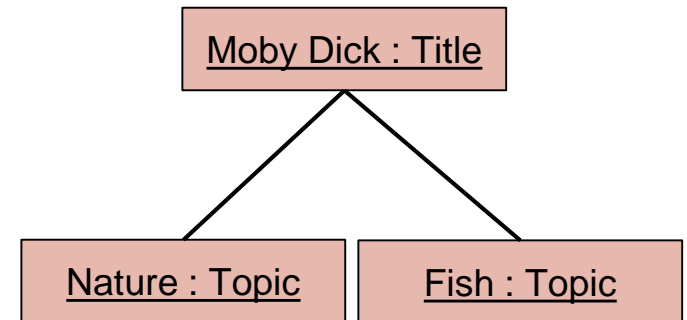
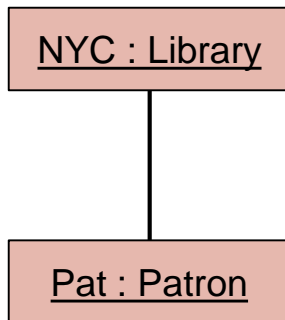
Actions

- Join Library (Patron)
- Search titles (Topic)

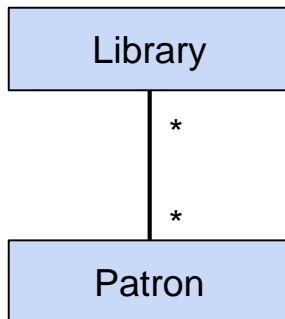
Notice our format:

- Underline actions
- Capitalize types

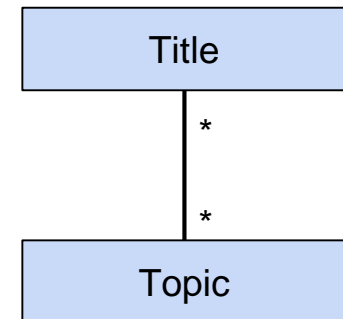
Instance



Type



The search yields titles
(i.e., books)



Scenario: **Library checks out Moby Dick**

Steps:

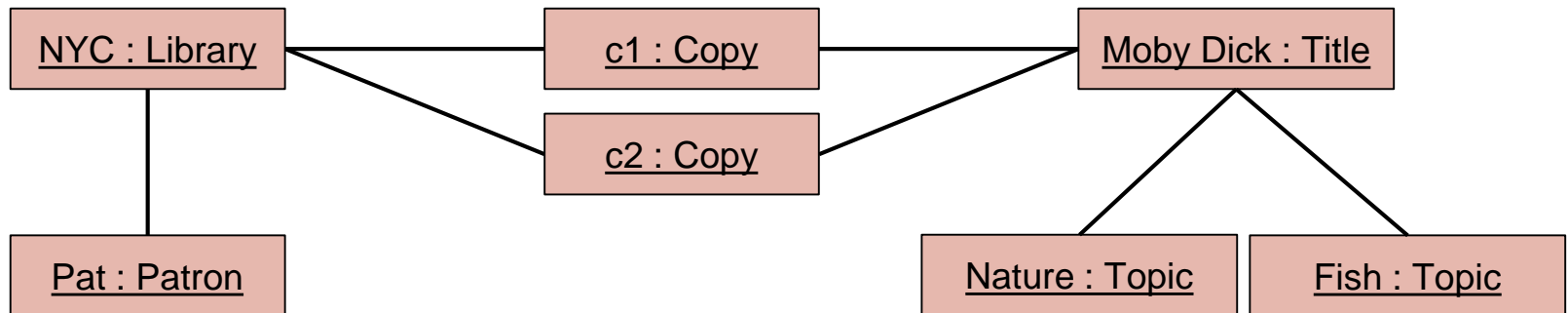
1. Pat (Patron) joins the NYC (Library).
2. Pat searches for titles about fish (Topic).
3. Pat checks out copy 2 of Moby Dick (Title).

Actions

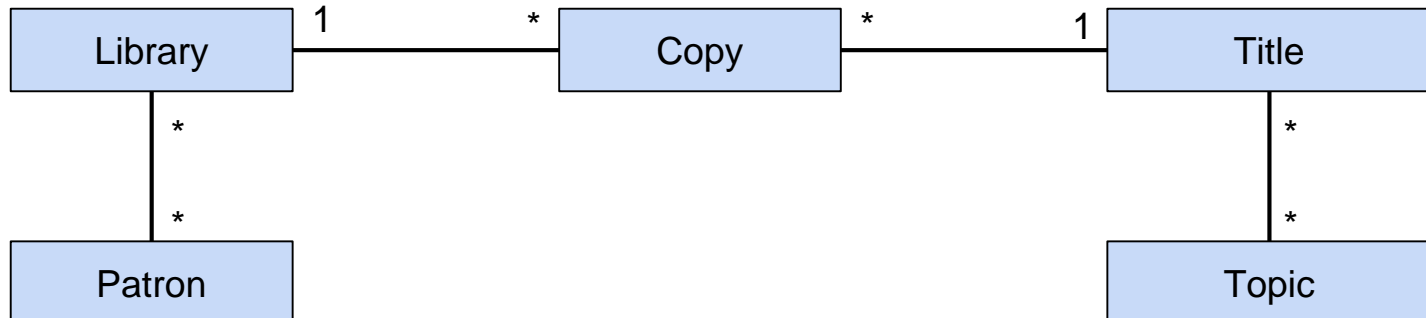
- Join Library (Patron)
- Search titles (Topic)
- Check out (Copy)

Next scenario step

Instance



Type



Scenario: **Library checks out Moby Dick**

Initial State:

- Copy 1 of Moby Dick is checked in.

Steps:

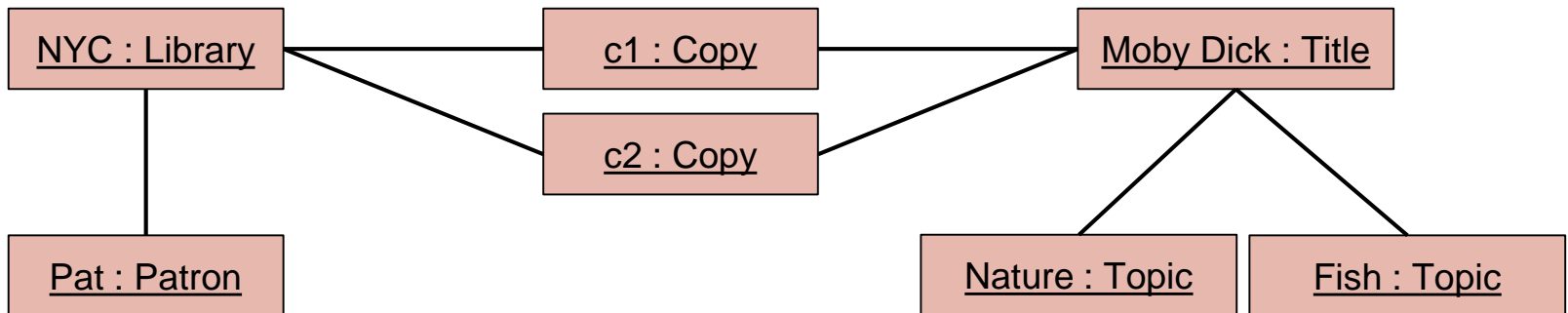
1. Pat (Patron) joins the NYC (Library).
2. Pat searches for titles about fish (Topic).
3. Pat checks out copy 2 of Moby Dick (Title).

Actions

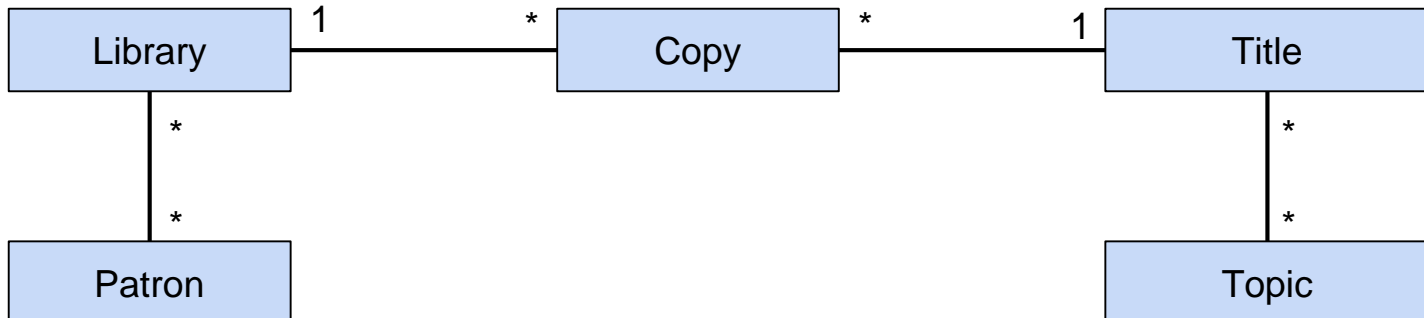
- Join Library (Patron)
- Search titles (Topic)
- Check out (Copy)

Can't check out a book that isn't checked in yet

Instance



Type



Scenario: **Library checks out Moby Dick**

Initial State:

- Copy 1 of Moby Dick is checked in.

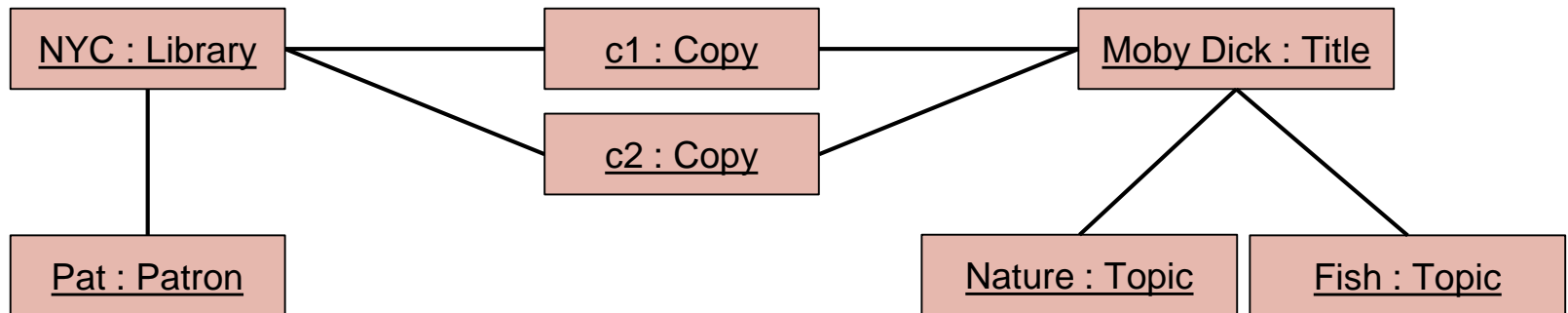
Steps:

1. Pat (Patron) joins the NYC (Library).
2. Pat searches for titles about fish (Topic).
3. Pat checks out copy 2 of Moby Dick (Title).
4. Pat returns copy 2 of Moby Dick.

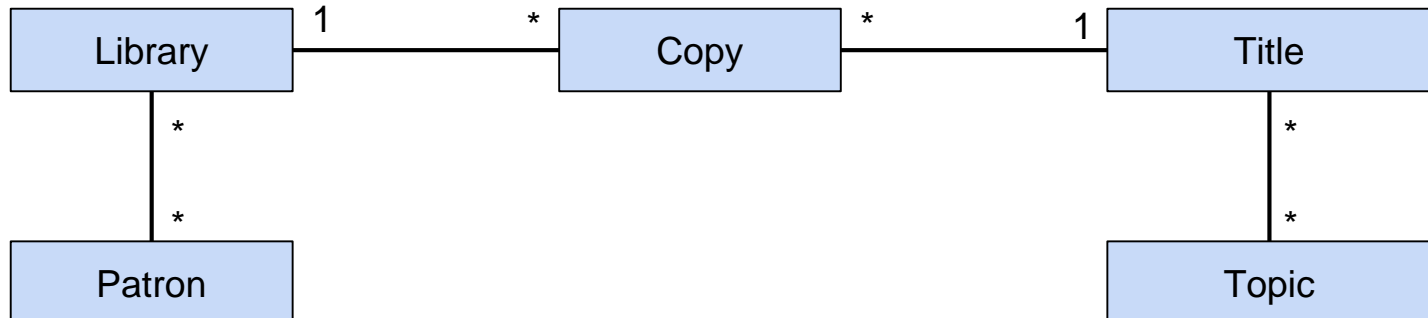
Actions

- Join Library (Patron)
- Search titles (Topic)
- Check out (Copy)
- Return (Copy)

Instance



Type



Behavior model

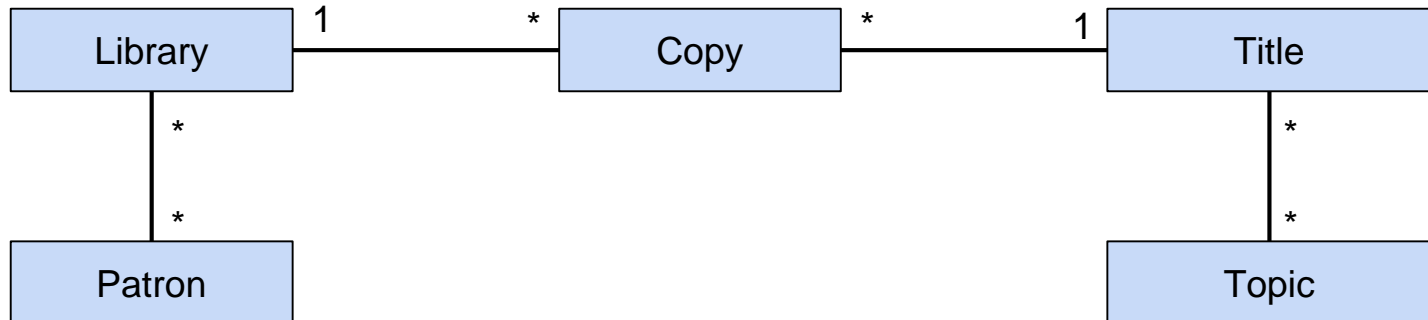
Minimally sufficient for the
scenario

Actions

- Join Library (Patron)
- Search titles (Topic)
- Check out (Copy)
- Return (Copy)

Information model

Minimally sufficient for the scenario



Type

Scenario: **Library checks out Moby Dick**

Initial State:

- Copy 1 of Moby Dick is checked in.

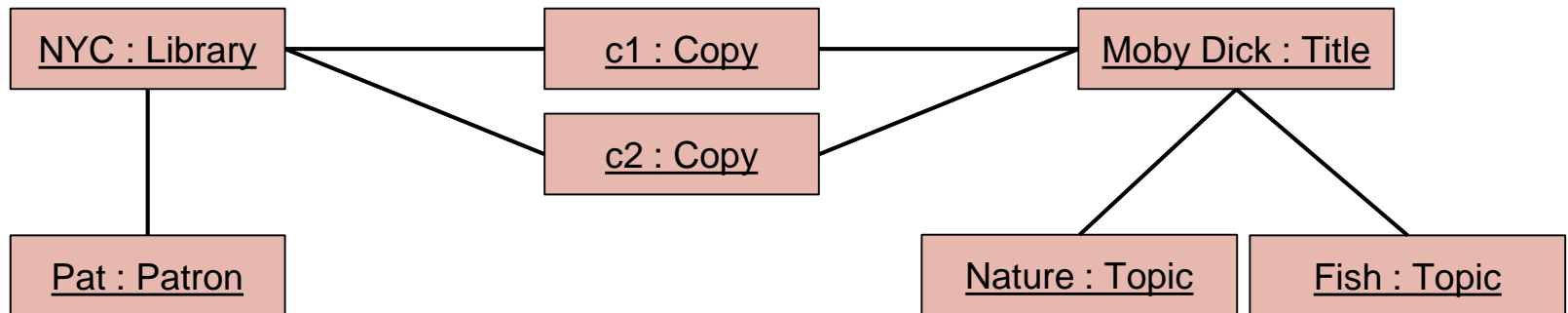
Steps:

1. Pat (Patron) joins the NYC (Library).
2. Pat searches for titles about fish (Topic).
3. Pat checks out copy 2 of Moby Dick (Title).
4. Pat returns copy 2 of Moby Dick.

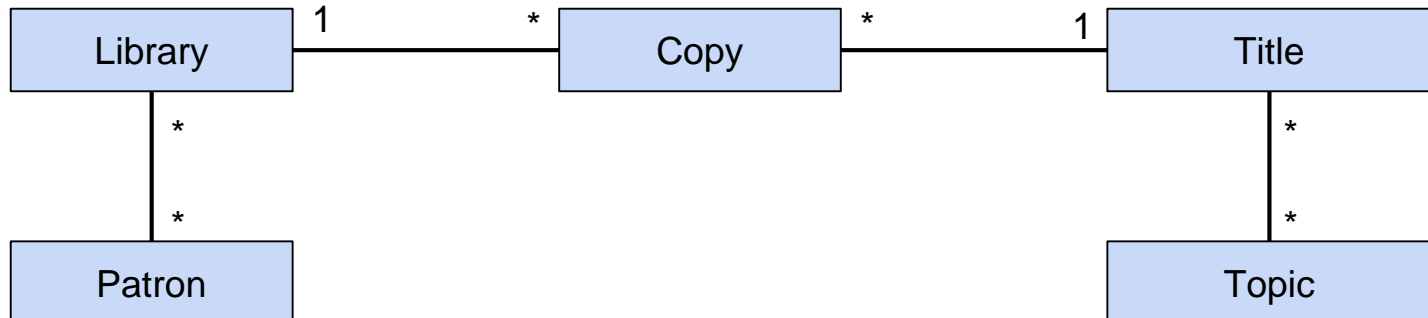
Actions

- Join Library (Patron)
- Search titles (Topic)
- Check out (Copy)
- Return (Copy)

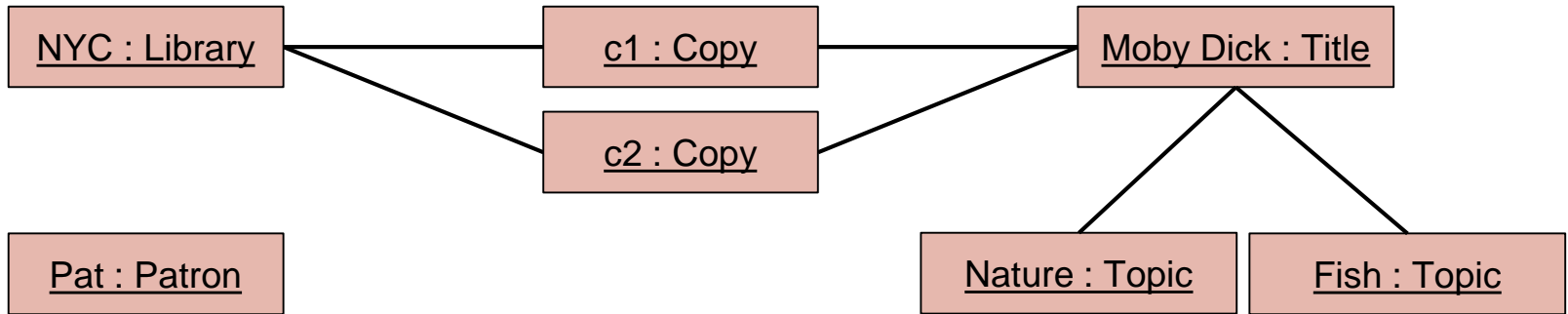
Instance



Type



Instance
(Before)



Scenario: **Library checks out Moby Dick**

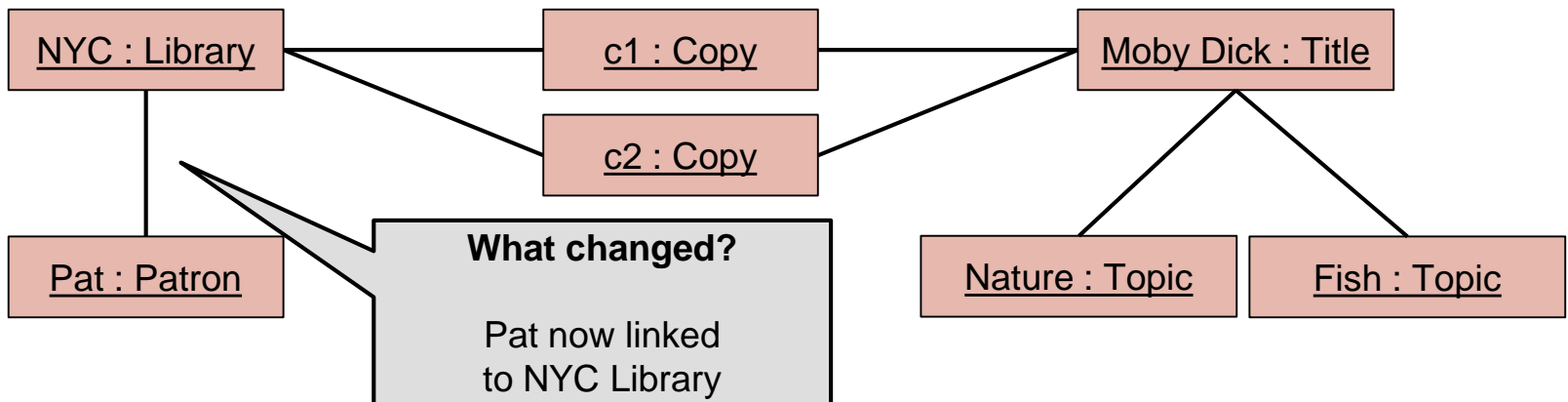
Initial State:

- Copy 1 of Moby Dick is checked in.

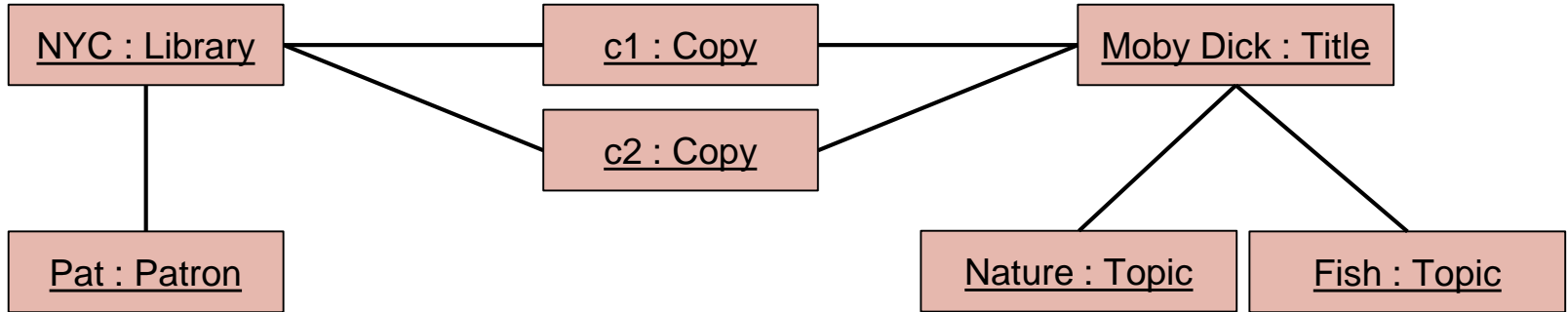
Steps:

1. **Pat (Patron) joins the NYC (Library).**
2. Pat searches for titles about fish (Topic).
3. Pat checks out copy 2 of Moby Dick (Title).
4. Pat returns copy 2 of Moby Dick.

Instance
(After)



Instance
(Before)



Scenario: **Library checks out Moby Dick**

Initial State:

- Copy 1 of Moby Dick is checked in.

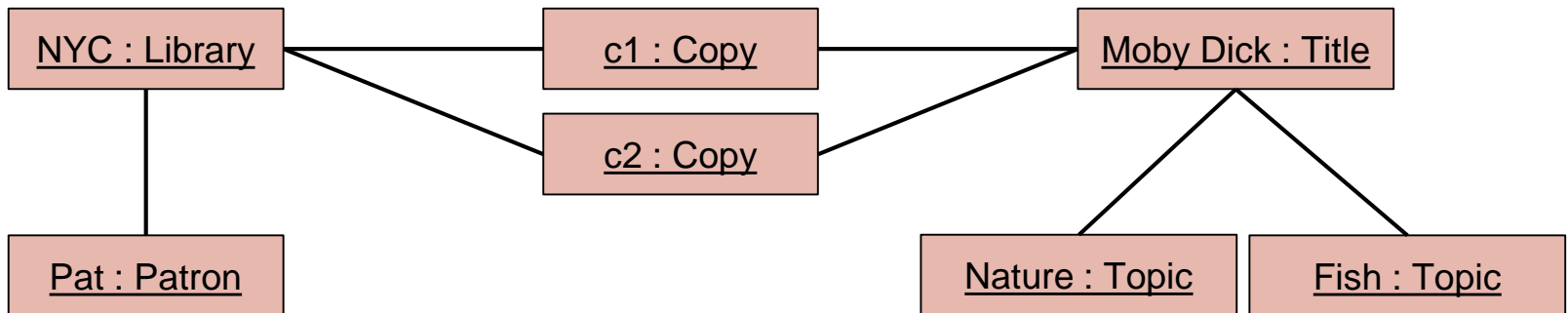
Steps:

1. Pat (Patron) joins the NYC (Library).
2. **Pat searches for titles about fish (Topic).**
3. Pat checks out copy 2 of Moby Dick (Title).
4. Pat returns copy 2 of Moby Dick.

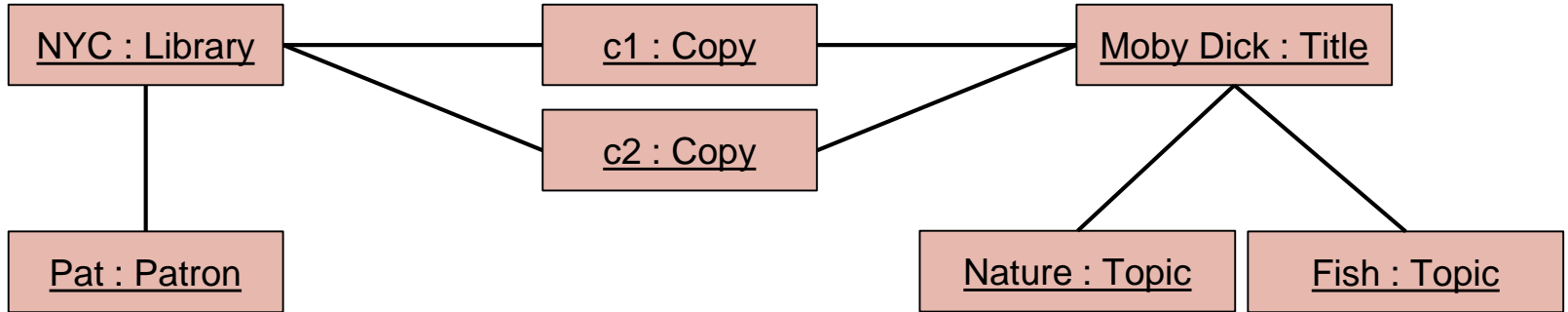
What changed?

Nothing. Probably OK for a search action.

Instance
(After)



Instance
(Before)



Scenario: **Library checks out Moby Dick**

Initial State:

- Copy 1 of Moby Dick is checked in.

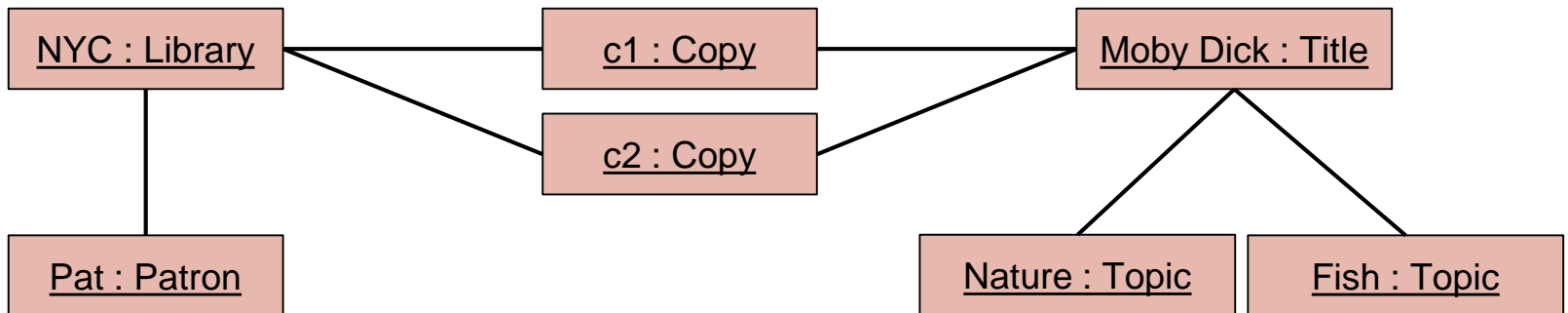
Steps:

1. Pat (Patron) joins the NYC (Library).
2. Pat searches for titles about fish (Topic).
3. **Pat checks out copy 2 of Moby Dick (Title).**
4. Pat returns copy 2 of Moby Dick.

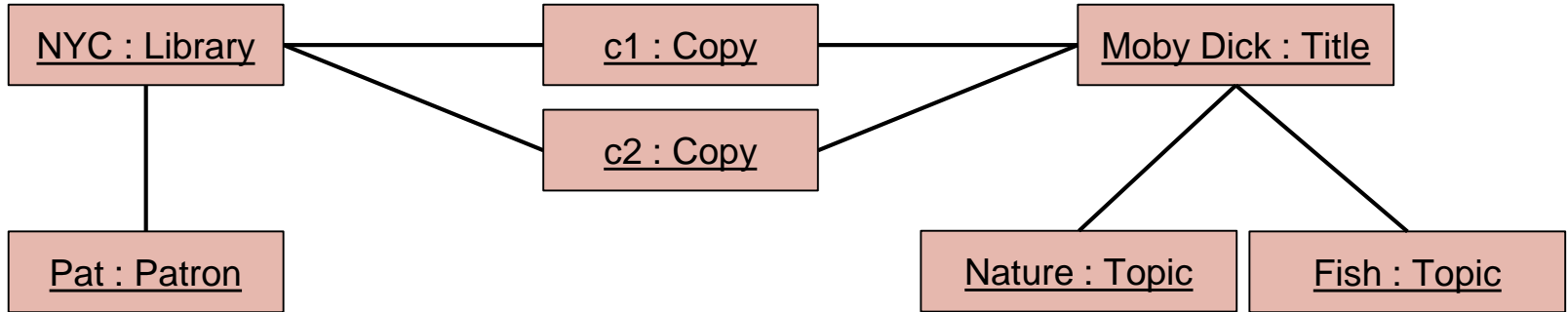
What changed?

Nothing. **NOT OK.**
Our model doesn't
explain the phenomena.

Instance
(After)



Instance
(Before)



Scenario: **Library checks out Moby Dick**

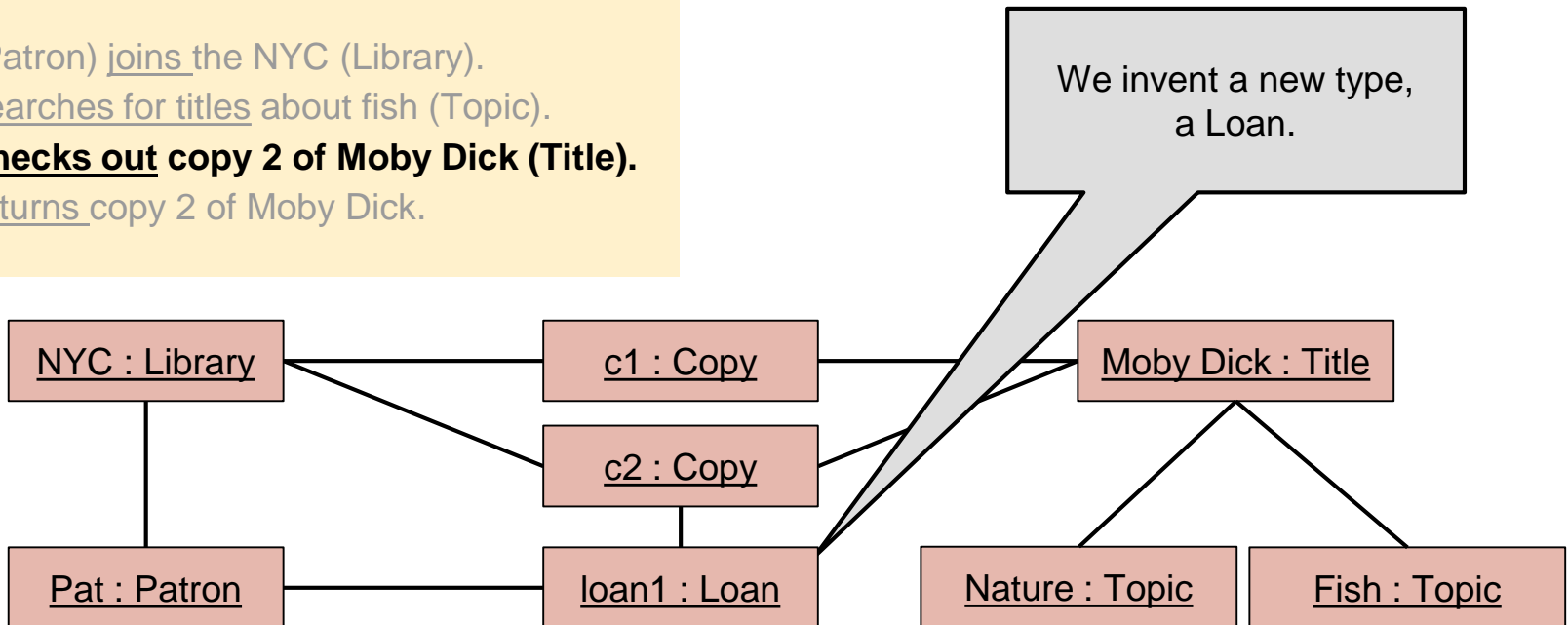
Initial State:

- Copy 1 of Moby Dick is checked in.

Steps:

1. Pat (Patron) joins the NYC (Library).
2. Pat searches for titles about fish (Topic).
3. **Pat checks out copy 2 of Moby Dick (Title).**
4. Pat returns copy 2 of Moby Dick.

Instance
(After)



Scenario: **Library checks out Moby Dick**

Initial State:

- Copy 1 of Moby Dick is checked in.

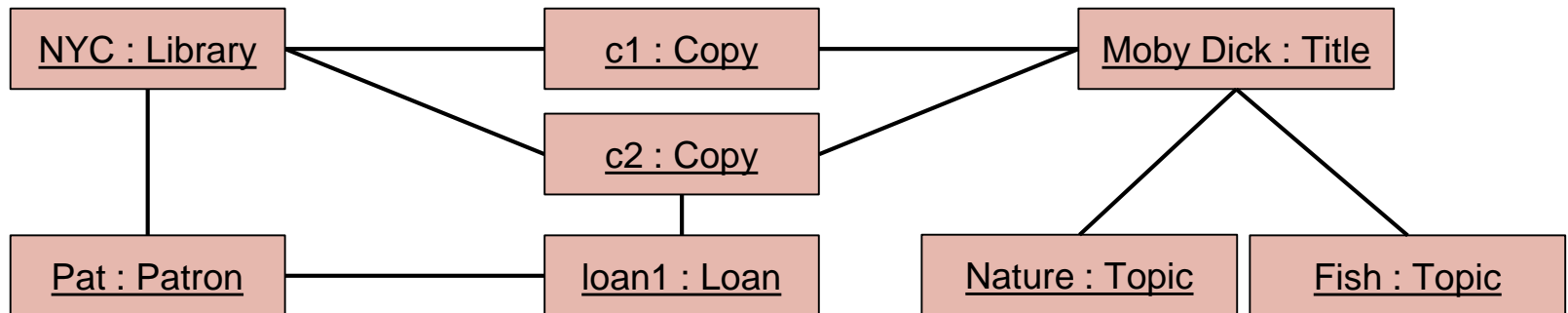
Steps:

1. Pat (Patron) joins the NYC (Library).
2. Pat searches for titles about fish (Topic).
3. Pat checks out copy 2 of Moby Dick (Title).
4. Pat returns copy 2 of Moby Dick.

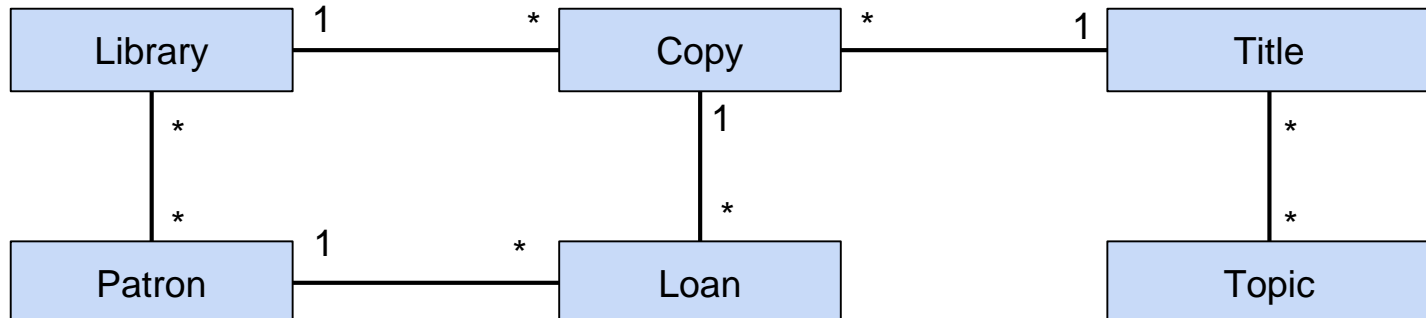
Actions

- Join Library (Patron)
- Search titles (Topic)
- Check out (Copy)
- Return (Copy)

Instance



Type



Scenario: **Library checks out Moby Dick**

Initial State:

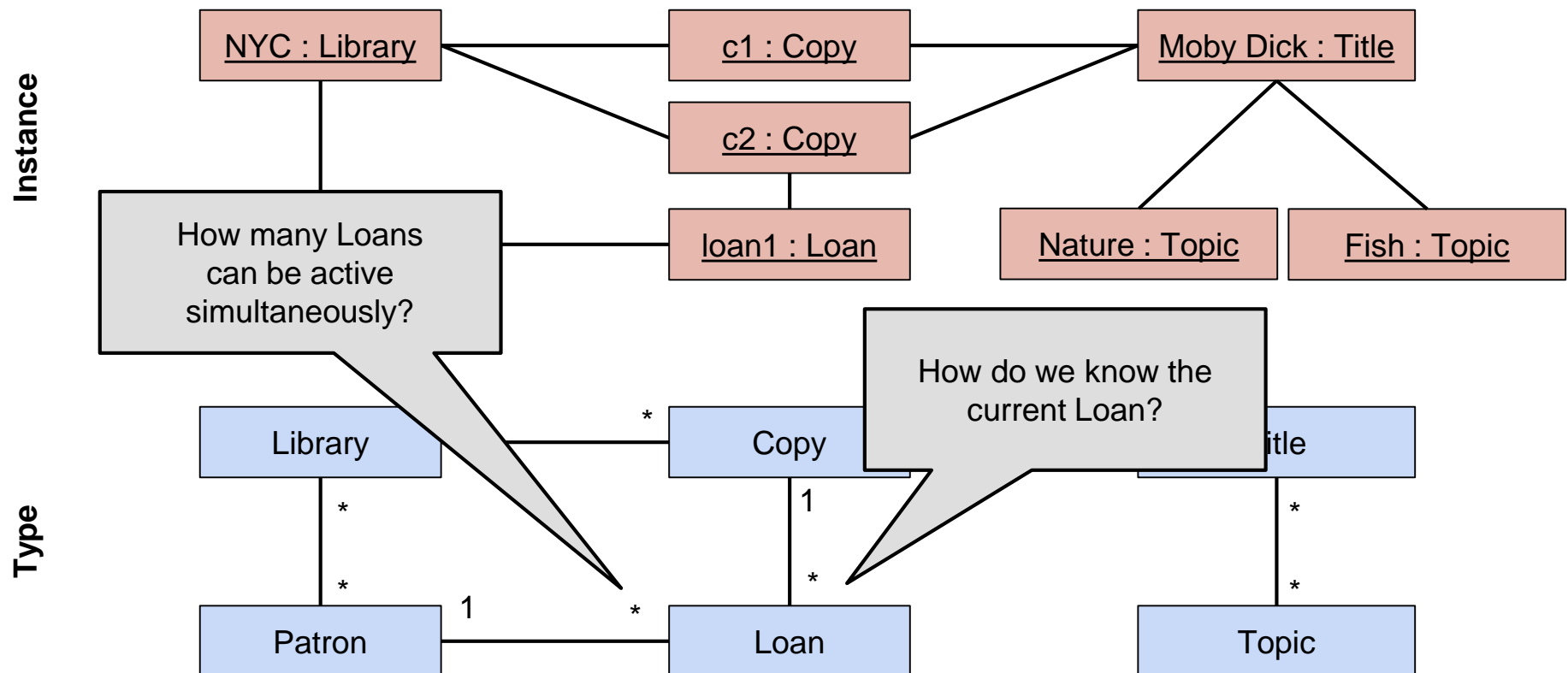
- Copy 1 of Moby Dick is checked in.

Steps:

1. Pat (Patron) joins the NYC (Library).
2. Pat searches for titles about fish (Topic).
3. Pat checks out copy 2 of Moby Dick (Title).
4. Pat returns copy 2 of Moby Dick.

Actions

- Join Library (Patron)
- Search titles (Topic)
- Check out (Copy)
- Return (Copy)



Scenario: **Library checks out Moby Dick**

Initial State:

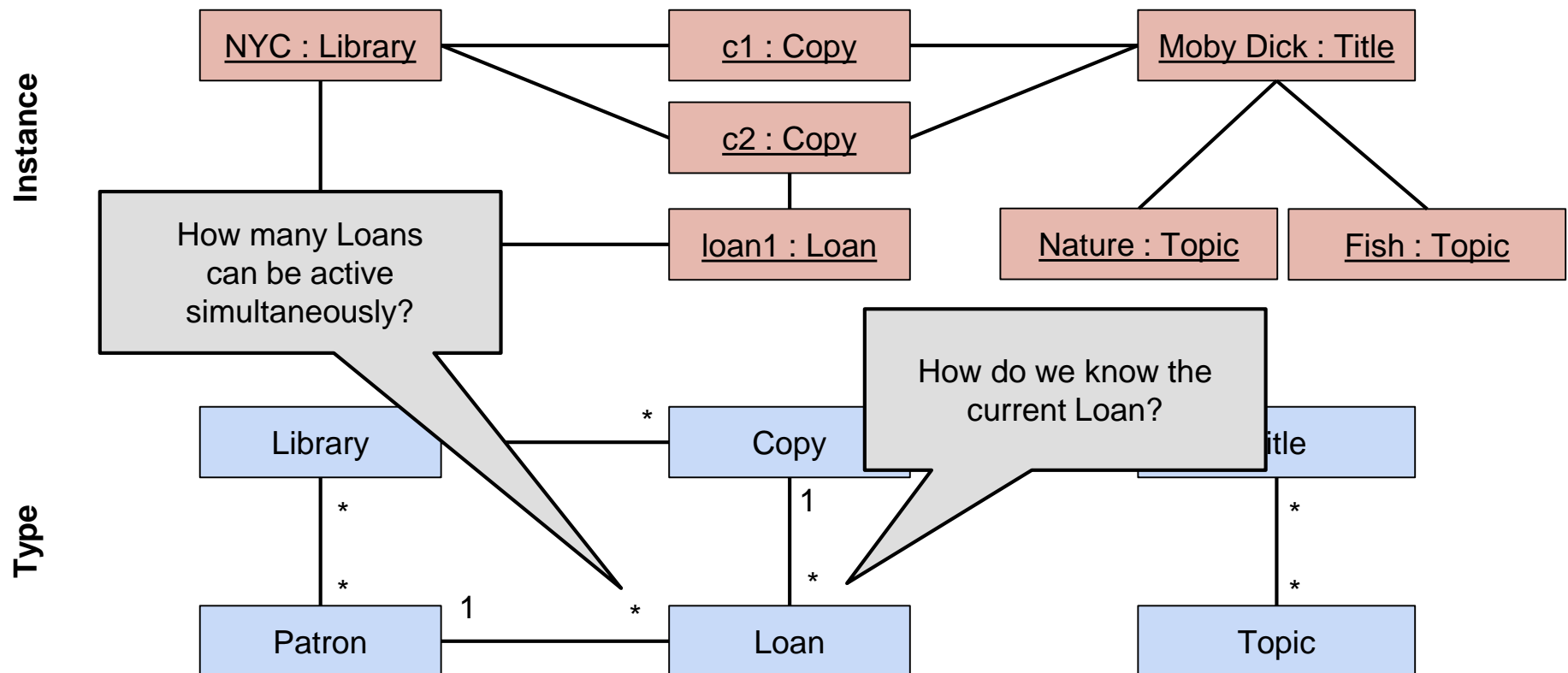
- Copy 1 of Moby Dick is checked in.

Steps:

1. Pat (Patron) joins the NYC (Library).
2. Pat searches for titles about fish (Topic).
3. Pat checks out copy 2 of Moby Dick (Title).
4. Pat returns copy 2 of Moby Dick.

Actions

- Join Library (Patron)
- Search titles (Topic)
- Check out (Copy)
- Return (Copy)



FIN.

Up Next...

Eric Willeke
Rally
@erwilleke

Mindsets over Methods (Eric's Silver Toolbox)

Eric Willeke
@erwilleke

Perspective matters

Mine: coach and change agent

Eric Willeke
@erwilleke

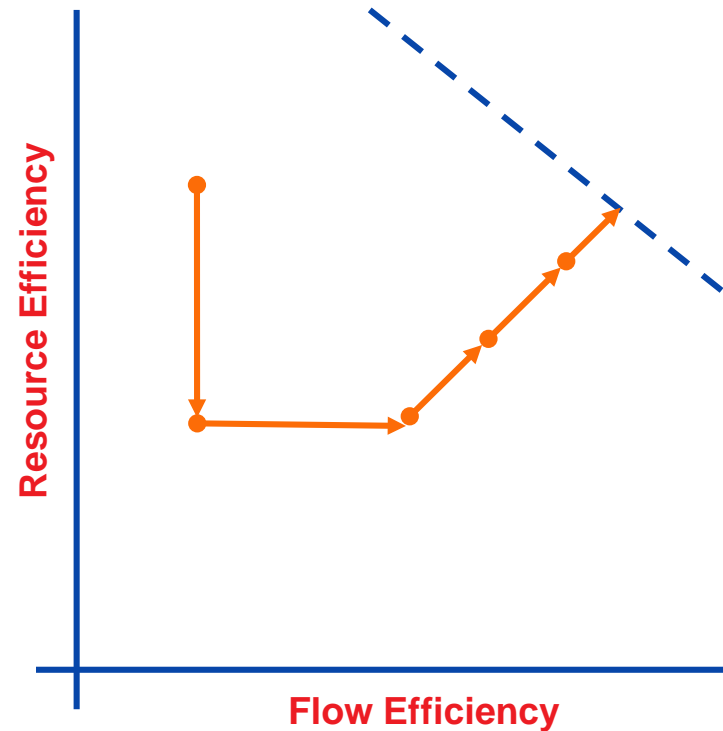
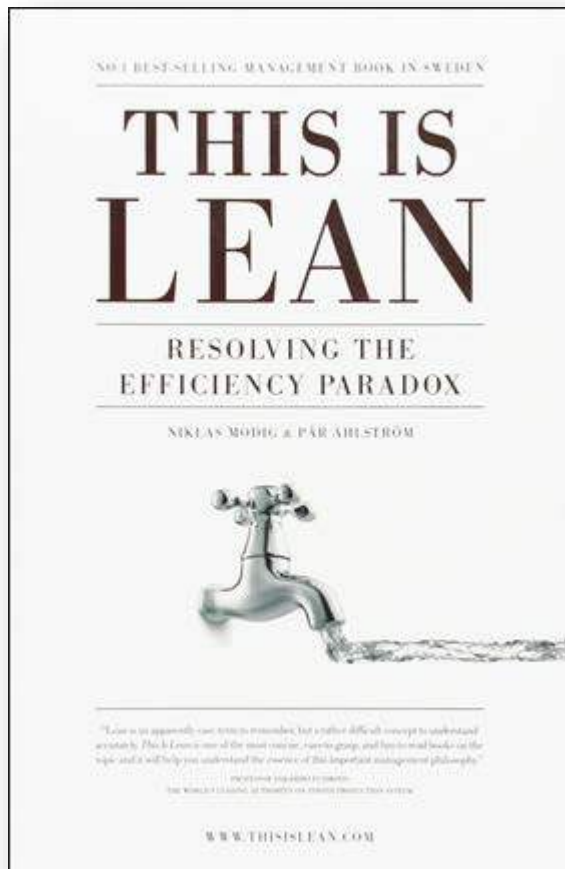
Principle-driven pragmatism

(Respect for Context)

Eric Willeke
@erwilleke

Methodologies are Countermeasures

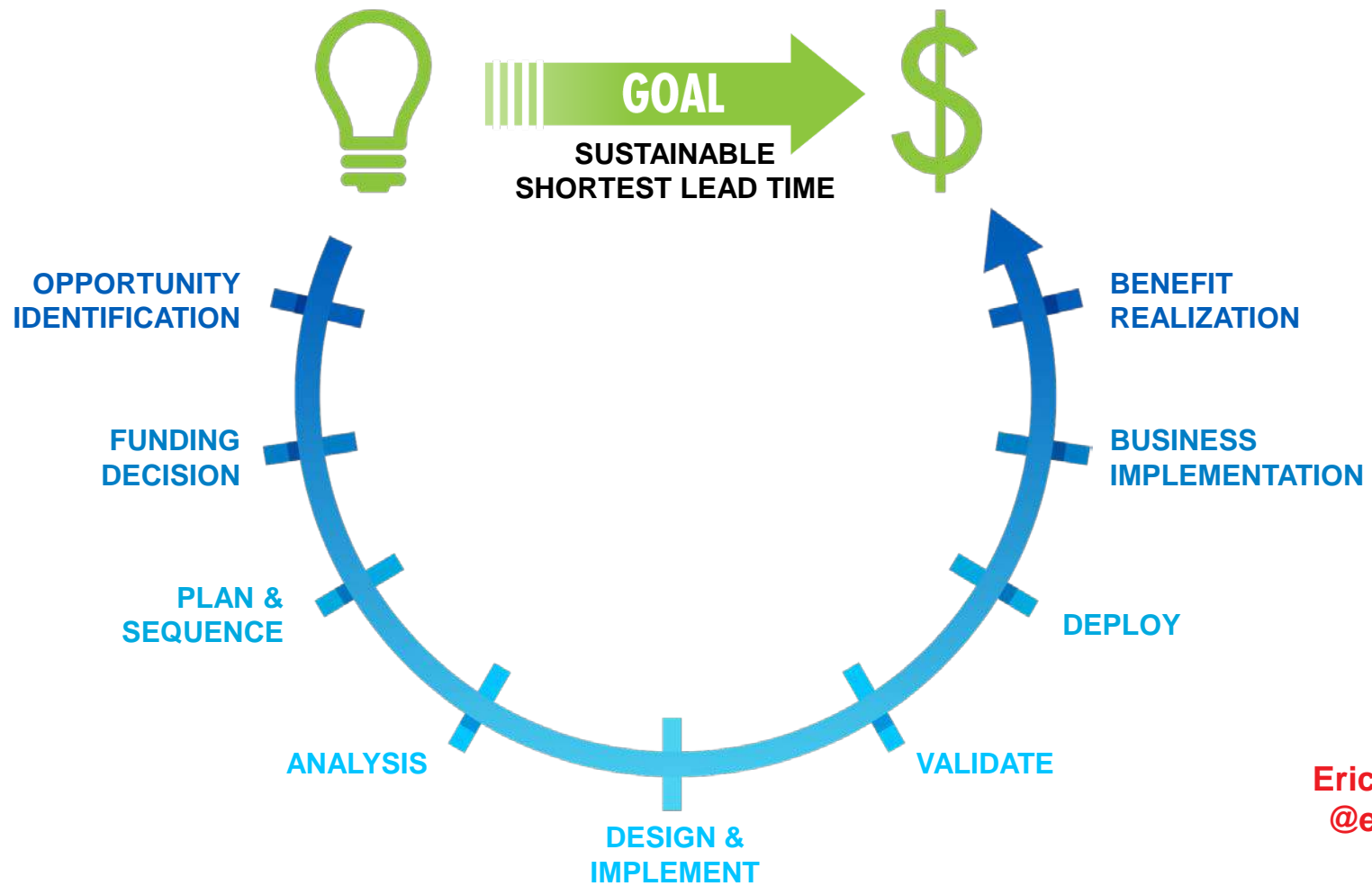
Eric Willeke
@erwilleke



Eric Willeke
@erwilleke

Outcome-driven Everything

Eric Willeke
@erwilleke



Eric Willeke
@erwilleke

Focus on the Bottleneck

Identify
Exploit
Subordinate
Elevate
Refocus

Eric Willeke
@erwilleke

People do better work in groups

Eric Willeke
@erwilleke

Networks trump Hierarchies

Eric Willeke
@erwilleke

Form Cross-functional Management Teams

Eric Willeke
@erwilleke

Recognize complexity... ... and simplicity

Eric Willeke
@erwilleke

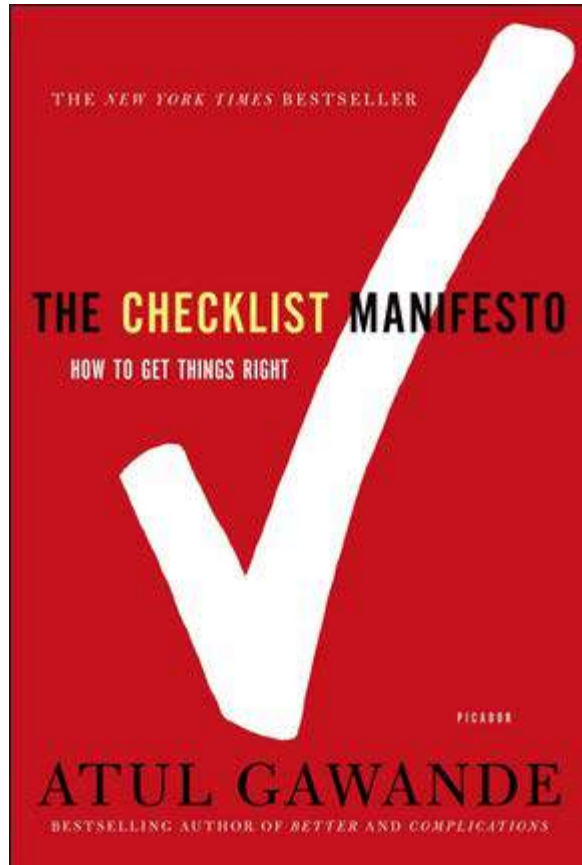
Blame-free bias

Eric Willeke
@erwilleke

Model in the moment

(Draw something!)

Eric Willeke
@erwilleke



Checklists enhance professionalism

Eric Willeke
@erwilleke

Directional Improvement

Eric Willeke
@erwilleke

**Perfect
and
Good Enough
are both the enemies of
Better**

**Eric Willeke
@erwilleke**

**All change is incremental,
define the intermediate steps**

Eric Willeke
@erwilleke

**Culture is the sum
of all behaviors**

Eric Willeke
@erwilleke

Mindsets over methods (Learn to see)

Eric Willeke
@erwilleke

FIN.

**WHAT'S IN YOUR
SILVER TOOLBOX?**

You've seen some ideas for what we have in our silver toolboxes.

What tools are in your
sliver toolbox?

- 3 rounds of collaborative brainstorming
- Not everything will fit into your toolbox
- No right or wrong answers
- Have fun!

Brainstorm as a pair

- Write down 2-3 ideas
- Use one sticky note per “tool”
- 3 minutes

Find another pair to
make a foursome!

Merge and Brainstorm

- Share ideas from Round 1
- You should have ~4-6 sticky notes
- 5 minutes

Prioritize your top 5 “tools”

Find another group to
make a group of eight!

Merge and Brainstorm Ideas

- Share ideas from Round 2
- You should have 10 sticky notes
- 7 minutes

Prioritize your top 5 “tools”
Silently!

What tools did you put
in your silver toolbox?

Silver Toolbox



Thank you!



Michael Keeling

@michaelkeeling

<http://neverletdown.net>

mkeeling@neverletdown.net