



Human-Centered Design + User Empathy

Week 3: Second Lecture

Real innovation comes from deeply understanding people

Product Design

Christopher A. Mattson, PhD
BRIGHAM YOUNG UNIVERSITY



**DESIGN
EXPLORATION
RESEARCH
GROUP**

Goal

Systematically **decompose** what we know about the users

Embrace iteration (informal and systematic)

Amp up our sketching

Outline

- **Business**
- **Out of Class Designette**
- **Empathy Maps & Journey Maps**
- **Revisioning (formalizing iteration)**
- **Sketching**
- **Closing**

	Su	M	T	W	Th	F	S	
Week 0								Design Philosophy & Engineering Design Essentials
Week 1								
Week 2								Complete Opportunity Development -- Initiate
Week 3					L		●	
Week 4		■	L	D	L		●	Complete Opportunity Development – User Insights
Week 5		■	L	D	L		●	
Week 6		■	L	D	L		●	Complete Concept Development
Week 7		■	L	D	L		●	
Week 8		■	L	D	L		●	
Week 9		■	L	D	L		●	Complete Sub-System Engineering
Week 10		■	L	D	L		●	
Week 11			L	D	L		●	Complete System Refinement
Week 12		■	L	D	L		●	Complete Product Pitch
Week 13		■	L					
Final Exam				●				Design Portfolio Due as Final Exam

Decomposition
Recombination (Table)
Most Critical Module/Interface
Geometry-Material-Process Decisions
Observations (+Thoughtless Acts)
Nucor's Circle (Hedgehog Concept)
5W Problem Framing Framework
Double Diamond Process
Conversation Starters
Design Thinking Process
Stages of Development
Design Thinking
How Might We Statements
Personas
Primary Research
Secondary Research
Empathy Map
Journey Map
Revisioning (formalizing iteration)

Designette

A Review of Your Work



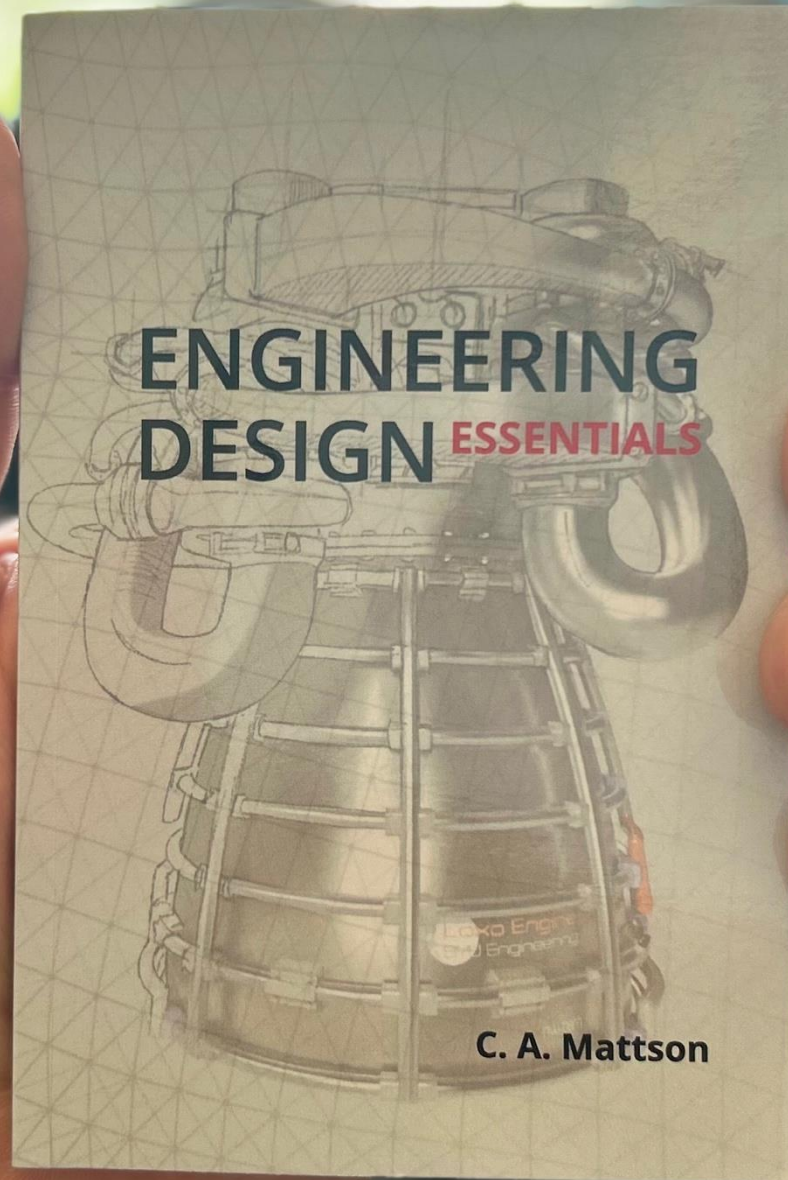
TEACH ME (I'm the Client who paid for the 6 observations, 3 pain points, and solution)

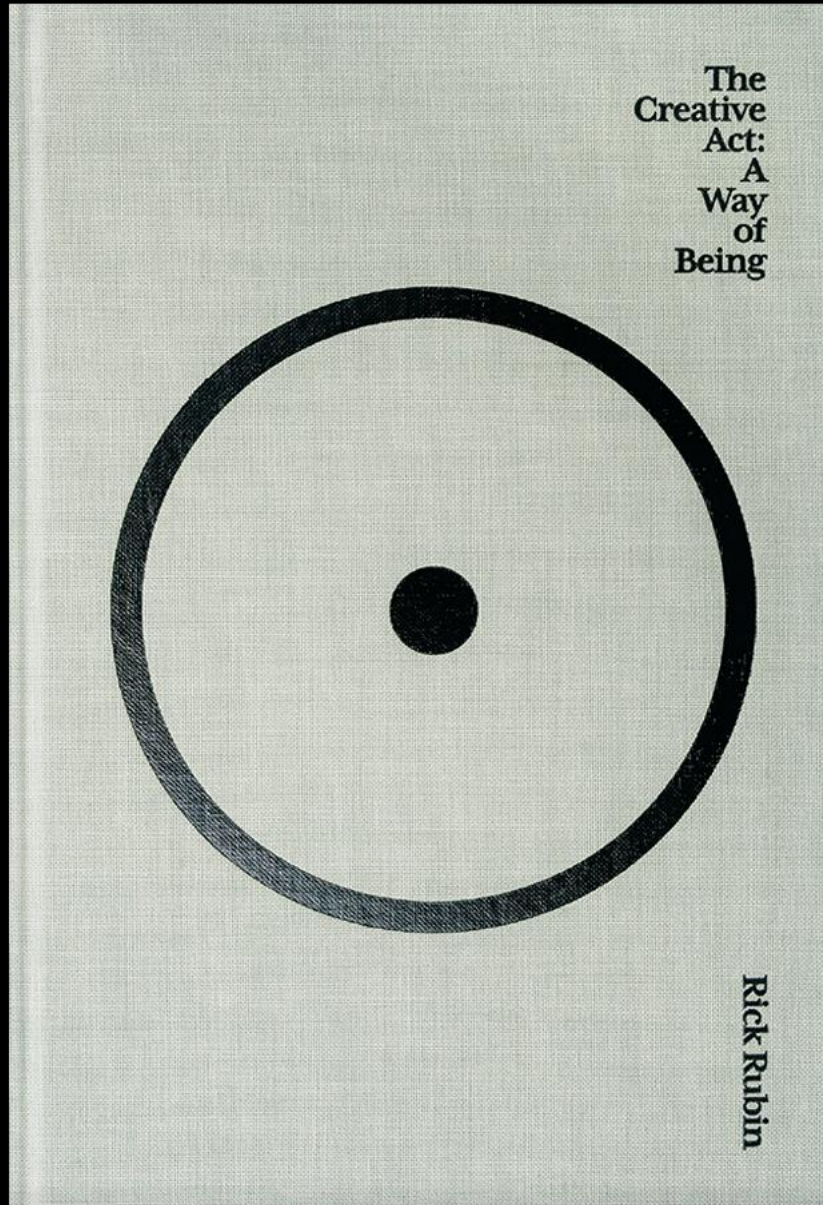
Purpose: Report as you would to a client

Task:

1. 1 minute: Gather and review your personal notes, insights, ideation
2. 5 minutes: Discussion of findings/solutions with small group (3 people) – be ready to present one observation, the pain, and proposed **engineering** solution
3. 1 minute: Verbal presentation to class (3 people will be chosen)

● Go to Page 90-91





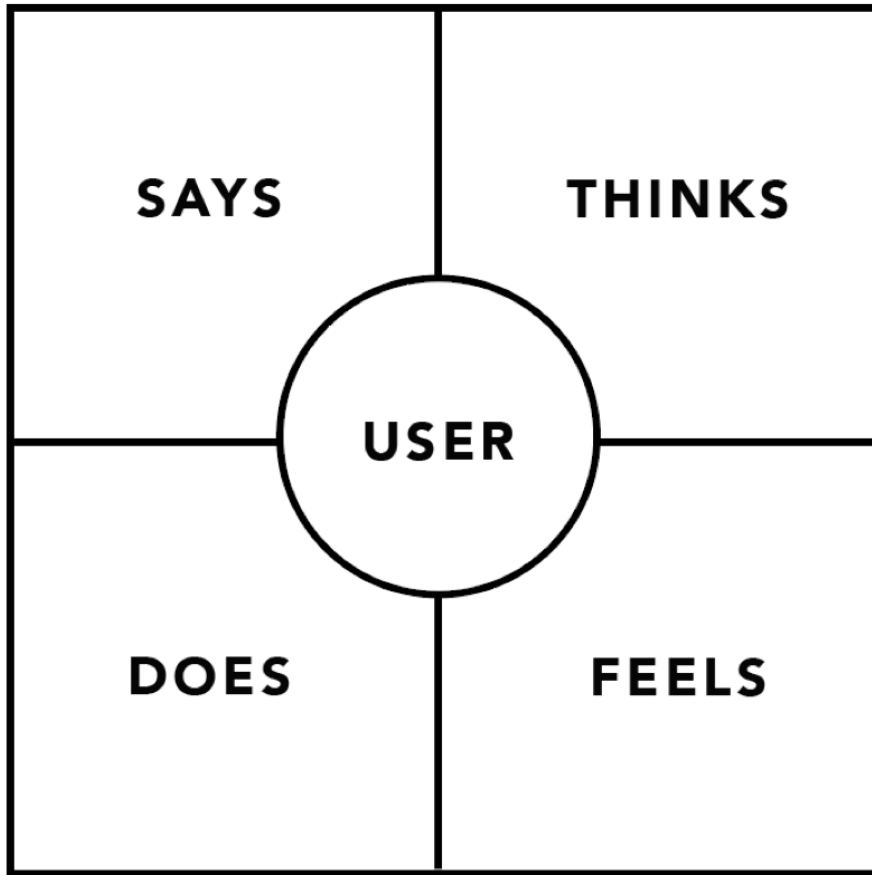
“The ability to look deeply is the root of creativity. To see past the ordinary and mundane and get to what **might otherwise be invisible.**”

Rick Rubin

Empathy Map

Decomposing the Human Experience (**emotional**)

Empathy Map



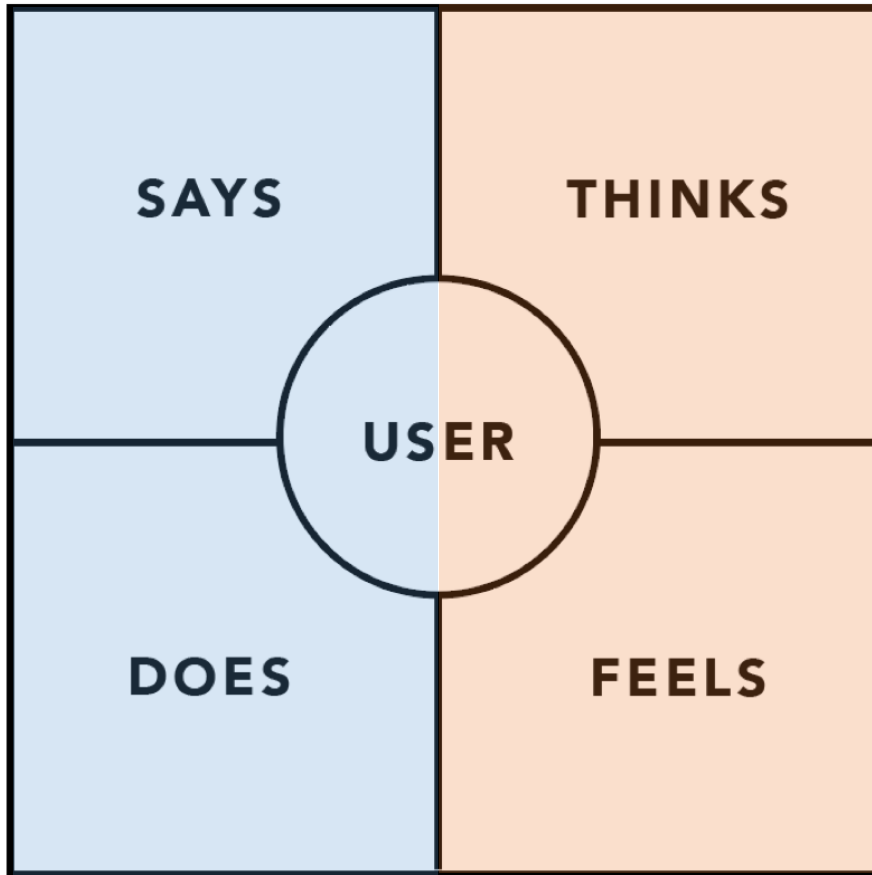
Definition (From NN Group)

An empathy map is a collaborative visualization used to articulate what we know about a particular type of user.

It externalizes knowledge about users in order to

- 1) create a shared understanding of user needs, and
- 2) aid in decision making.

Empathy Map



Definition (From NN Group)

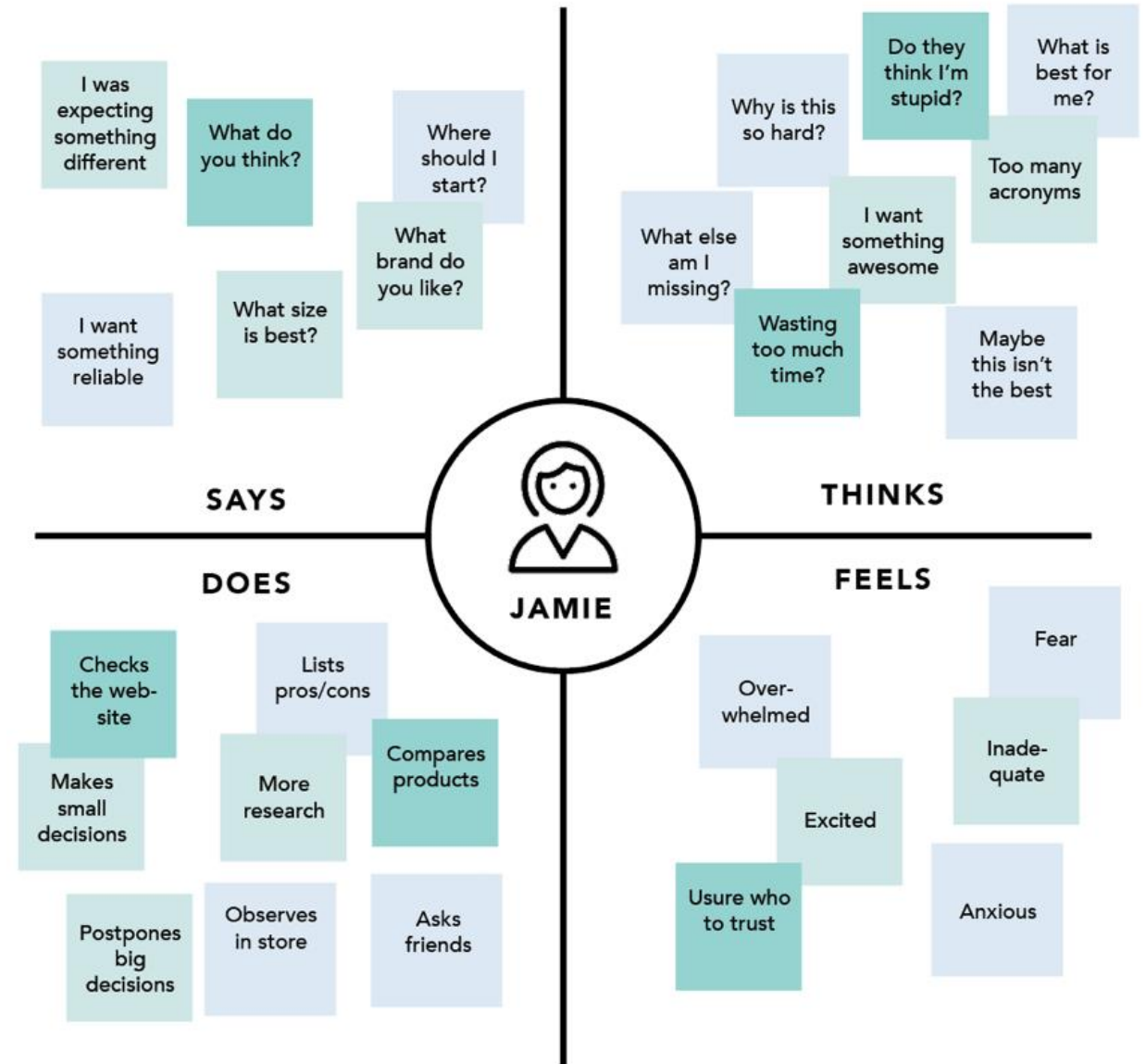
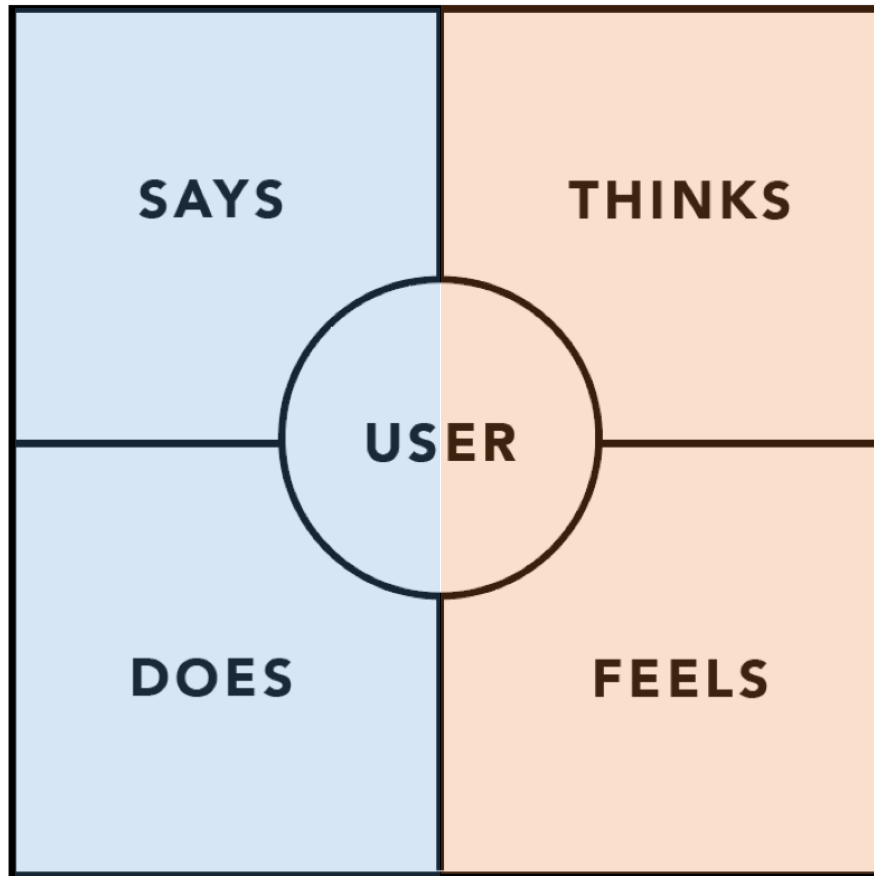
An empathy map is a collaborative visualization used to articulate what we know about a particular type of user.

It externalizes knowledge about users in order to

- 1) create a shared understanding of user needs, and
- 2) aid in decision making.

Empathy Map

Example (Buying a TV)



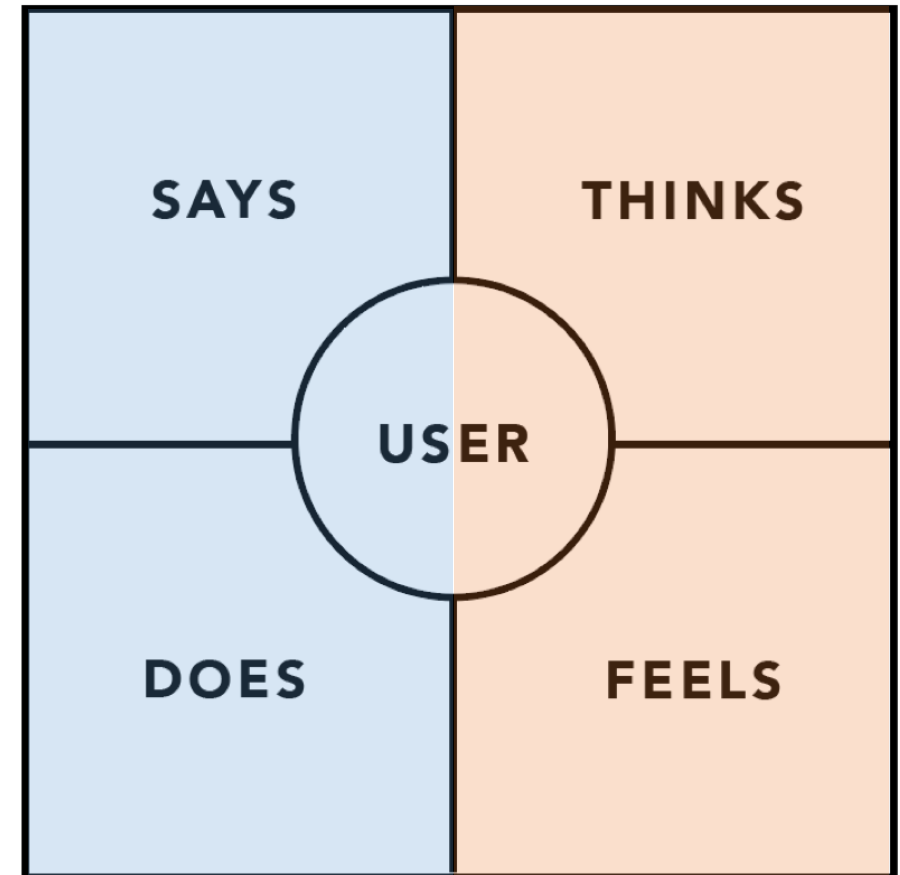
ACTIVITY (5 minutes)

Purpose: Create a preliminary empathy map for your design project

Task:

1. Draw the box (:
2. Choose a user
3. Write words in each sub box to articulate what you **know**.

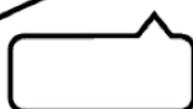
You have about 60 seconds per box



Journey Map

Decomposing the Human Experience (**physical**)

Journey Map

 SPECIFIC USER + SCENARIO + GOALS		
PHASE 1	PHASE 2	PHASE 3
1. _____ 2. _____ 	3. _____ 4. _____ 5. _____ 6. _____ 	7. _____ 8. _____ 9. _____ 
OPPORTUNITIES + INTERNAL OWNERSHIP		

Definition (From NN Group)

A journey map is a visualization of the process that a person goes through in order to accomplish a goal.

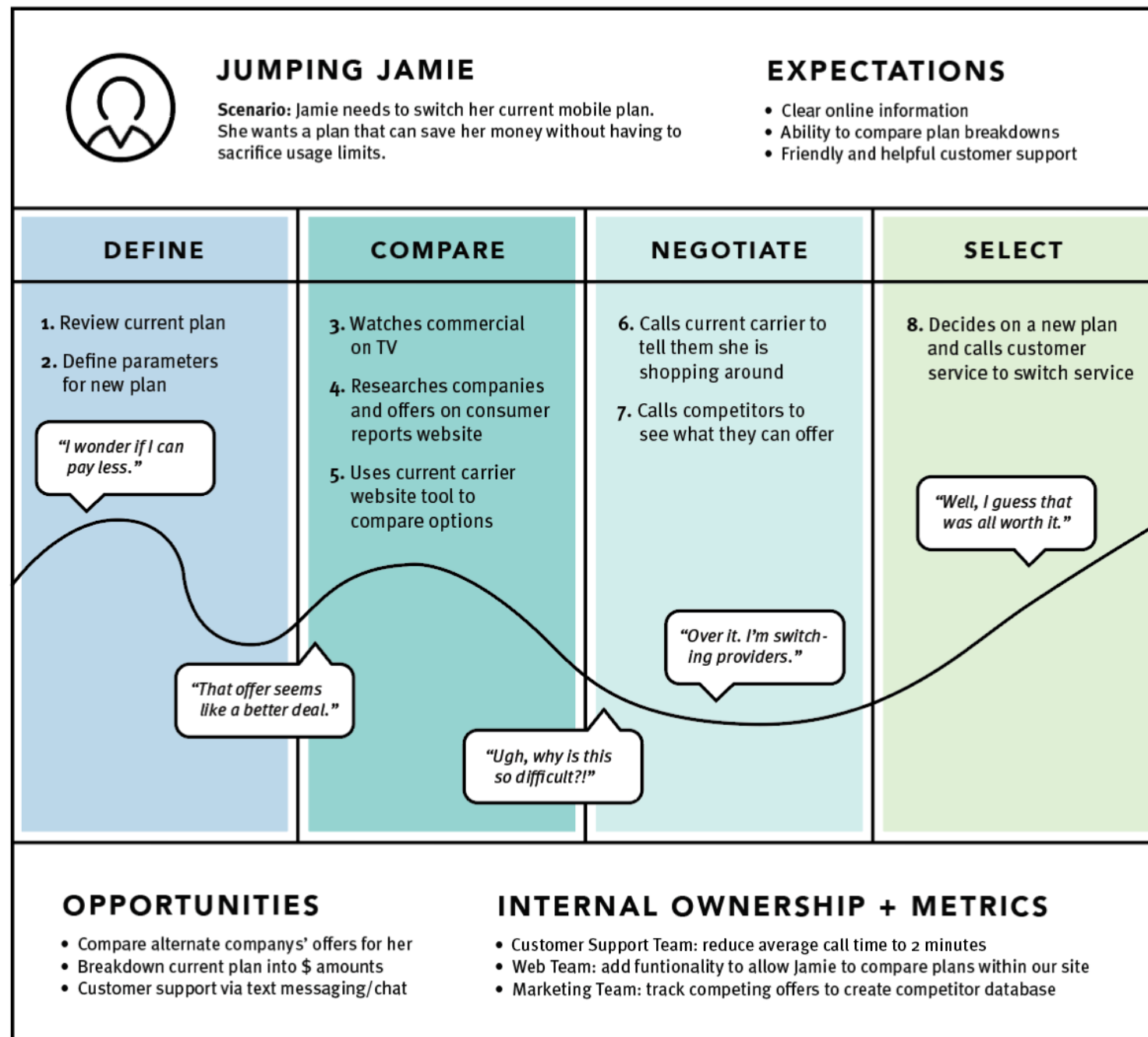
It is a series of actions on a timeline, which also captures thoughts and emotions.

It is a time-based representation of the empathy map, revealing operational and emotional sequencing.

Journey Map

Example

Switching Mobile Phone Plans



Journey Map

Example

Switching Mobile Phone Plans

Scenario & Expectations →

Journey Phases →

User Actions →

User Thoughts →

User Emotions →

Design Opportunities →



JUMPING JAMIE

Scenario: Jamie needs to switch her current mobile plan. She wants a plan that can save her money without having to sacrifice usage limits.

EXPECTATIONS

- Clear online information
- Ability to compare plan breakdowns
- Friendly and helpful customer support

DEFINE

1. Review current plan
2. Define parameters for new plan

"I wonder if I can pay less."

COMPARE

3. Watches commercial on TV
4. Researches companies and offers on consumer reports website
5. Uses current carrier website tool to compare options

"That offer seems like a better deal."

NEGOTIATE

6. Calls current carrier to tell them she is shopping around
7. Calls competitors to see what they can offer

"Over it. I'm switching providers."

"Ugh, why is this so difficult?!"

SELECT

8. Decides on a new plan and calls customer service to switch service

"Well, I guess that was all worth it."

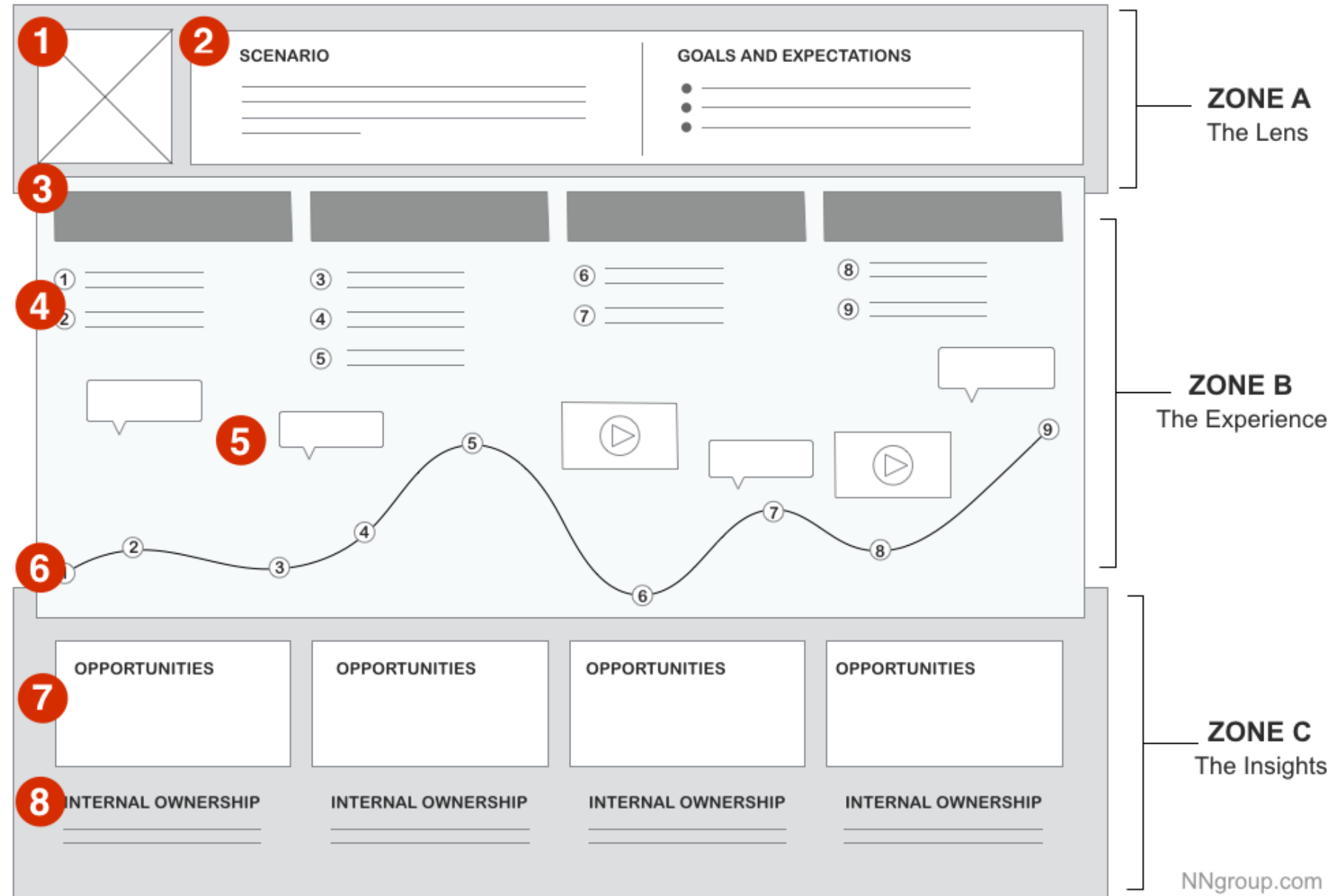
OPPORTUNITIES

- Compare alternate companys' offers for her
- Breakdown current plan into \$ amounts
- Customer support via text messaging/chat

INTERNAL OWNERSHIP + METRICS

- Customer Support Team: reduce average call time to 2 minutes
- Web Team: add funtionality to allow jamie to compare plans within our site
- Marketing Team: track competing offers to create competitor database

1. Stakeholder
2. Scenario & Goals
3. Journey Phases
4. Stakeholder Actions
5. Stakeholder Thoughts
6. Stakeholder Emotions
7. Design Opportunities
8. Ownership for Design





Sarah

Scenario

Sarah is going to the movies. She is excited to go out for the night and will meet her friend at the theater.

Expectations

- Great movie
- Friendly staff
- Good seat

Decide

1. Looks up movies on phone
2. Decides which movie to see and which theater to go to
3. Buys ticket online from phone

"I wonder if I can find a closer theater."

Travel

4. Drives to movie theater
5. Stands in line and buys popcorn
6. Finds seat next to friend

"Traffic is always so bad!"

"Where is my friend?!"

Experience

7. Watches movie

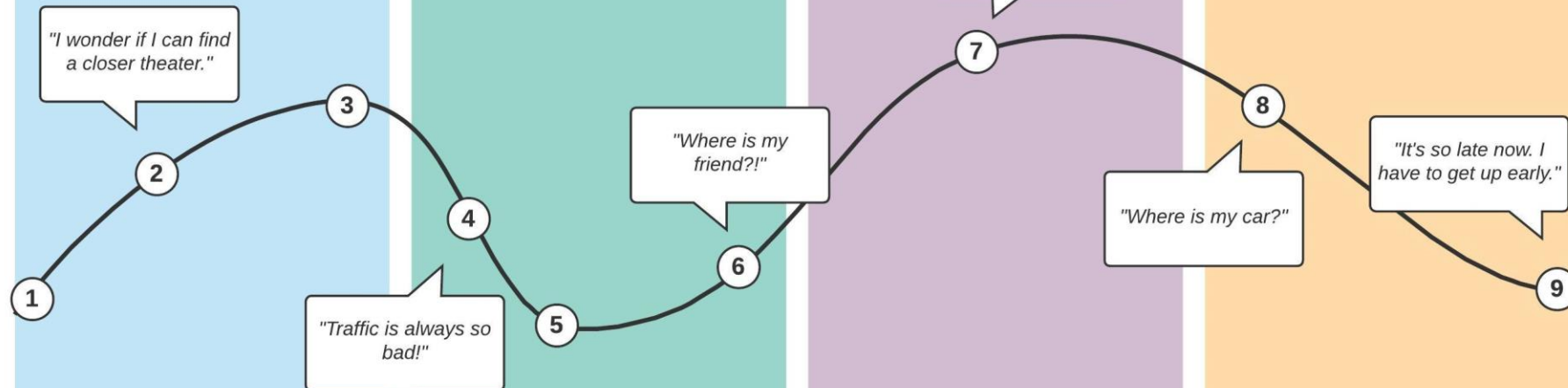
"Finally I can relax."

"Where is my car?"

Return

8. Exits theater
9. Drives home

"It's so late now. I have to get up early."



Themes

- Overall process for deploying code is a blend of manual and automated processes, and takes a long time. It can be easy for a developer to get distracted and miss things.
- Developers consistently look to other team members who are more experienced with the shipping process. Documentation is assumed to be out-of-date or hard to find, so people rely on their team to answer questions.
- Docker configuration is not consistent, causing issues that require troubleshooting from various team members. Building Docker images can be a lengthy process, and the base image is not ideal.
- It can be unclear what part of the process Infrastructure owns versus developers.
- There is a learning curve to understanding Shippy vs Ship-It, and which is preferred.
- Logs are hard to find in Kibana and include few instructions, making the debugging process frustrating and problematic.
- There is no clear process to escalate issues, and developers resort to randomizing the Infrastructure team for assistance.
- There is a desire to learn how to improve, yet teachable moments are missed when "fires" are put out or fixed quickly and developers move on to other issues.
- Onboarding is perceived to be a difficult, days-long process without clear guidance or trustworthy reference material.

Development

Verification

Deployment

DOING



Developer codes



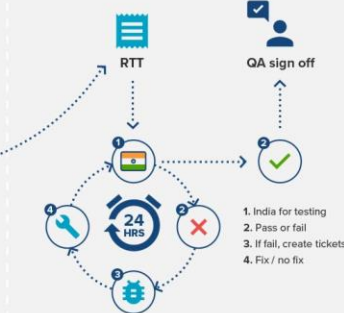
Code review



Automated tests

- Create a pull request. Commit code changes.
- Test code locally.
- Look at other apps for guidance.
- I don't have permission to set up docker build, have to contact Infrastructure.
- Need clarity and don't have a reference; I'll turn to someone I know and trust who has knowledge (expert or mentor).
- For bigger Infra work, I have to create a Jira ticket and follow up. Might take a few weeks to get unblocked.

- Create a pull request and have peers on my immediate team check it out. Slack devs to say, "hey, look at this."
- Link Jira tickets into a release.
- Newer devs get review from mentors, then improve code.
- More experienced devs get reviews sooner/more frequently.
- When tests fail, restart and they usually pass.
- Interrupt Infrastructure team: "start a fire" to get attention in #infra.



- Spin up an EC2 (background task that takes about an hour).
- **X Spin up 2 EC2s for redundancy, in case one goes down in the night.**
- Throw away the EC2 and spin up a new one when running into blocking issues.
- Ask Infra if there are environment issues.
- Coordinate with India test team.

1. India for testing
2. Pass or fail
3. If fail, create tickets
4. Fix / no fix

X Antipattern



Open Ship-It



Create ticket



In queue

- Open Ship-It.
- Go to an expert for Ship-It advice.
- Start a new ship ticket and get in the queue.



Babysitting



Stag → Prod



- Sitting around waiting for things to happen.
- Finish shipping at home.
- Missing the 5 minute window.
- Not big on documentation -- go to #coolkids if there's a problem.

1. Automated tests run
2. Pass or fail
3. If fail, create log
4. 5 mins. to fix
5. If miss 5 min. window

- Turn to the internets for help.
- Try to track down something mysteriously wrong (was it my code or the environment?).
- Turn to mentor for help.



Go live



Check site



Honeybadger

- Monitor to make sure it went out smoothly.
- If it fails, manually roll back and monitor everything for the next few days.
- "Fall forward."

THINKING

- Docker is still mysterious to me.
- Process is inconsistent and not well defined. Who do I talk to? Don't know where to go.
- No good docs. Documentation "rots." Confluence formatting is not easy to use for code.
- Estimating timelines: small changes seem to take 1 day dev, 2 days RTT, 1 day ship.
- Not sure about the scope of what Infrastructure owns -- I'll learn about it through trial and error.
- Making a new service is much more ambiguous than working on an existing service. There's no single approach or template for how to do it.

- "Pull requests are the foundation of shipping."
- I'm going to test locally a lot before coming close to shipping.
- Rancher deploys get stuck on various problems.
- "Am I ready to go?"
- Transient failures are common in automated tests.

- Each EC2 is a snowflake -- nothing is the same twice.
- Some things are only testable in some environments. Not everything works on EC2, but it's unknown until the situation arises.
- Hard to tell if it's an environment problem or code problem when something goes wrong.

- Rancher on EC2 has improved. More stable recently.
- Front end is harder to test because of browser issues and communication with India team.

- I have to make sure I do things in the right order.
- This could take hours, depending on what's ahead of me in the queue.
- Seeing the visual shipping pipeline is nice.
- Ship-It can be super fast.
- Shippy is awesome for automation and best practices -- encourages you to take the right steps. It's nice and straightforward after learning it.

- Can take hours depending on what's ahead in the queue.
- Blockers and failures are common.
- Text notification is helpful.
- "I don't trust that when Shippy said something when wrong, it actually went wrong."

- Transient failures are common.
- Deploy is rickety. You run into problems that aren't your own, "restart tests and they usually pass." Unreliable. Multiple single points of failure.

- I didn't know rollback is an option, and I don't know how to do it -- I'll ask an expert.

FEELING

- Uncertainty.
- Setting up environment is frustrating.
- Docker config is a pain and slow.
- "My job is not to be a devops wizard."

- Documentation is outdated and unhelpful, but I am supported by my immediate team.

- Setting up environment is frustrating.
- RTT process feels like "friction."
- Debugging environment issues is difficult and sometimes impossible.

- Ship-It makes it sound scarier than it really is. There's more to how Ship-It works than I understand.
- "I'm terrified of shipping"
- Excited to ship. I want to get it done but I feel the need to be cautious.
- "I just need to get this out" and "it should just work."
- The easiest path to take is the **wrong** path during shipping.
- Confused about Ship-It vs Shippy, when to use project vs. queue, hidden menus in Ship-It, and the "button I'm not supposed to press."

- This is like waiting in line at the DMV.
- Cross my fingers and hope it will work.
- Hard to find logs (frustrating, this is a pain, "scattered").

- Kibana syntax is hard with no instructions. Can't find where the problem is because I don't know what to look for.
- Honeybadger is helpful.
- Troubleshooting EC2 is frustrating.

- It feels really good when you actually ship something!
- Not confident when rolling back.

OPPORTUNITIES ROOTED IN PAIN POINTS

1 Clear process for questions and requests.

Make it clear and easy to know the path to escalate problems or get work items prioritized by Infrastructure. Make it feel safe and welcoming to ask questions and make requests. Educate devs on what Infrastructure can help with.

2 Essential documentation.

Documentation isn't helpful for everything, but sometimes references are needed. Focus on creating and maintaining a few essential documents that are always up-to-date and trustworthy. For most questions, continue to rely on the strong mentor relationships that already exist on the dev team.

3 Confidence in the tools.

Find ways to make the **right** path the **easiest** path, make the UI straightforward and easy to understand, and remove unnecessary noise from the tool.

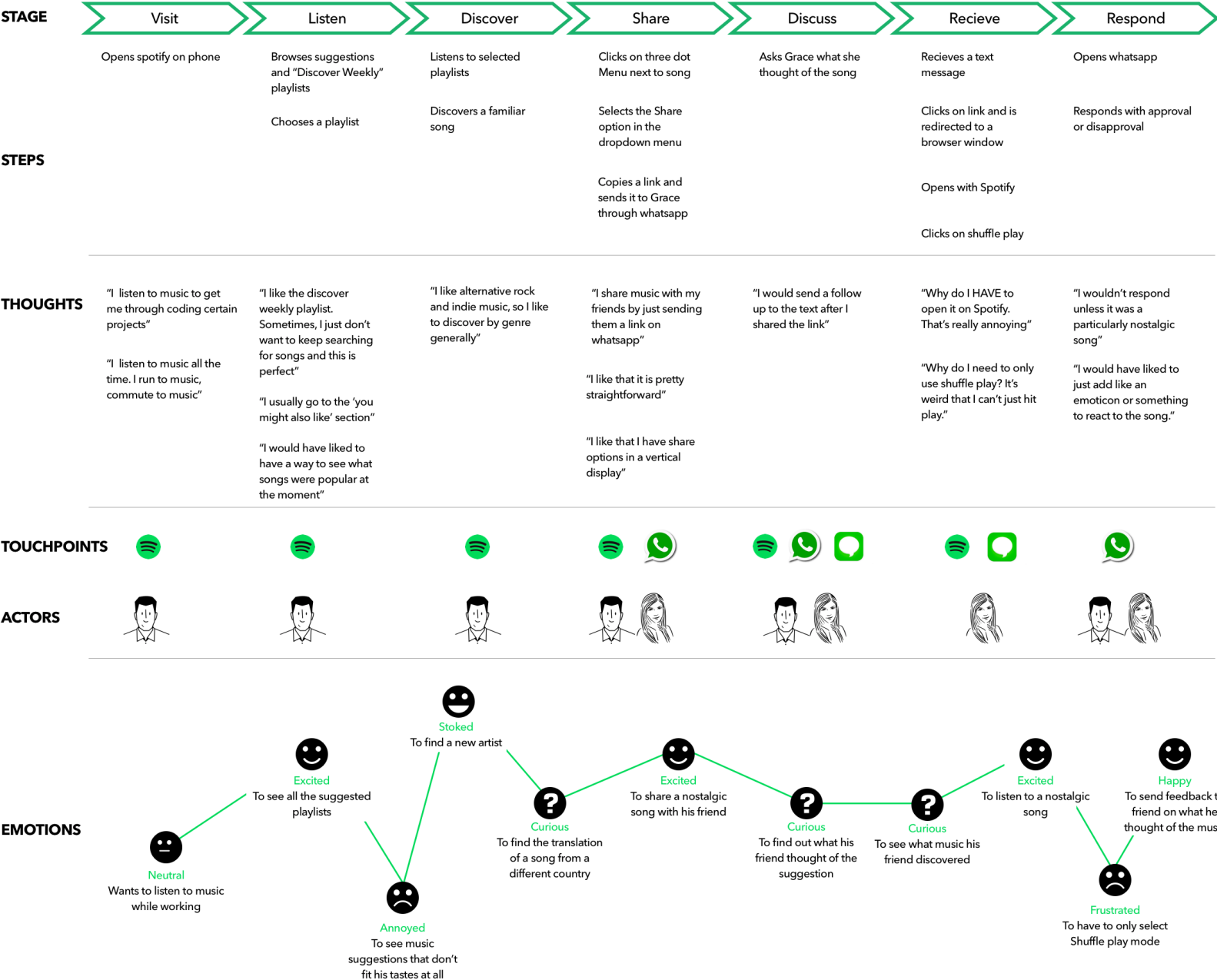
4 Easier shipping queue.

Help set expectations for how long something might take, or give devs more control over their wait. For times when a hands-on response is needed or the shipper is distracted, find ways to remove the hassle of having to start over. Look for more automated, time-saving solutions, like text message notifications.

5 Excitement boost when you ship.

Shipping is exciting! It feels great when something ships successfully. Boost this feeling of accomplishment with delightful interactions, subtle rewards, or positive messaging.

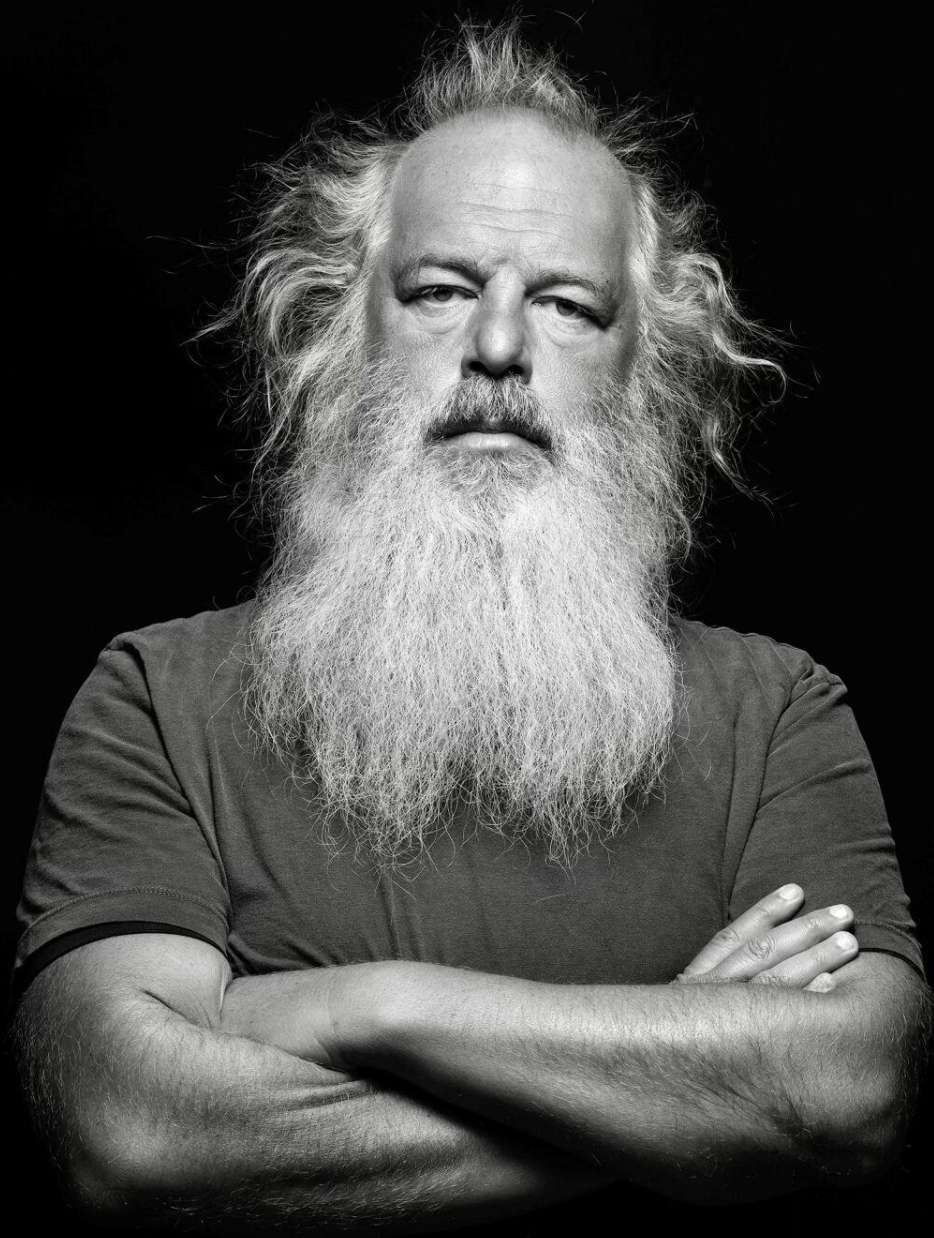
Customer Journey Map - Music Sharing Experience



DESIGN

Think deeper about Design
Talk more intelligently about Design
More intentionally **practice** Design

**then you
ever have before**



“The ability to **look deeply** is the root of creativity. To **see past the ordinary** and mundane and get to what might otherwise be invisible.”

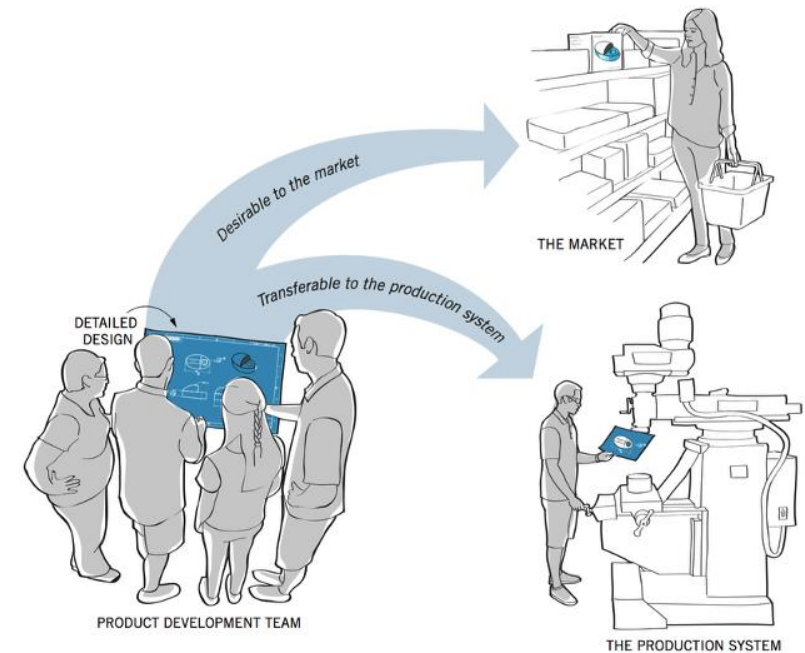
Rick Rubin
Music Producer

Revisioning

Formalizing Iteration

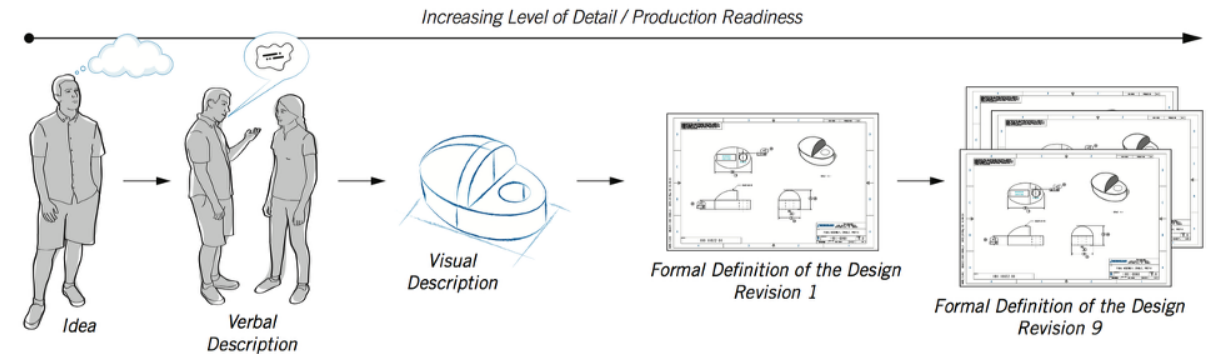
Principle 1

The job of the product development team is to create a design that is desirable and transferable

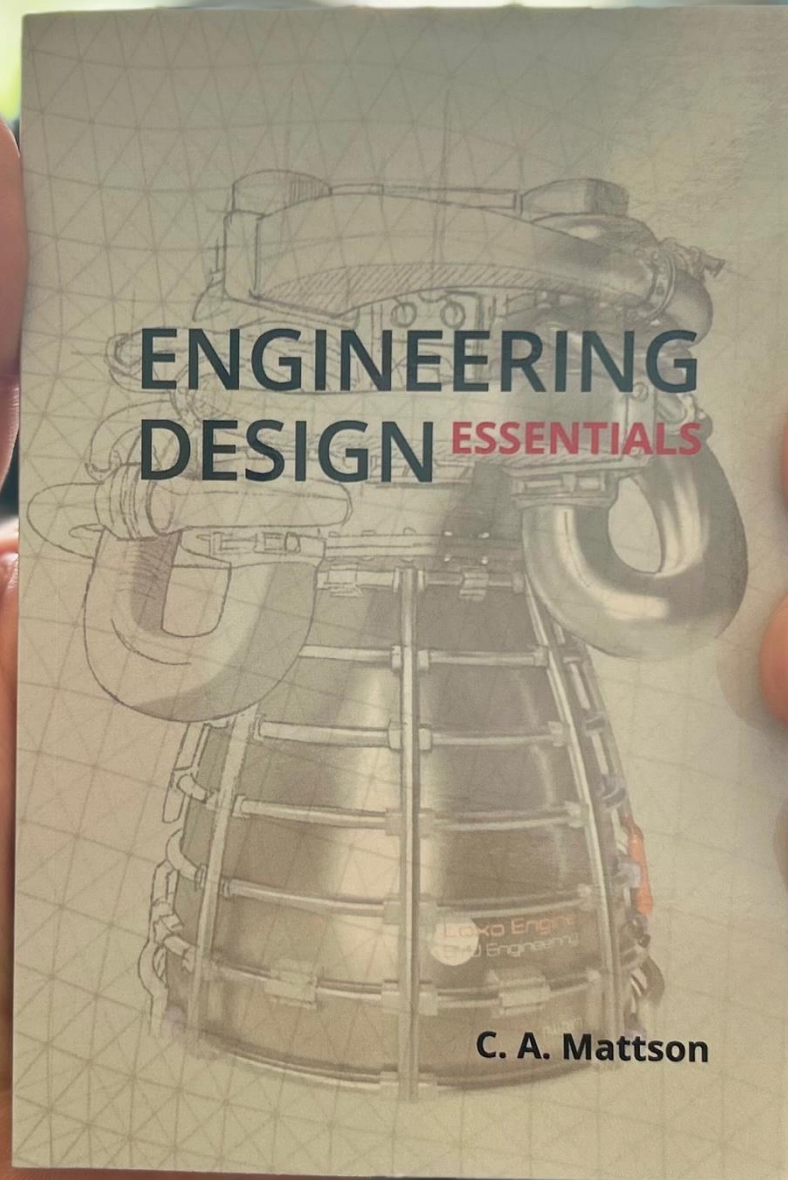


Principle 2

The design must evolve, gradually becoming better, more mature, and until it contains all the necessary information for the production system to manufacture the product and test its quality



● Go to Page 79



Domain Eval and Selection_V0.1.pdf
Domain Eval and Selection_V0.2.pdf
Domain Eval and Selection_V1.0.pdf
Domain Eval and Selection_V1.1.pdf

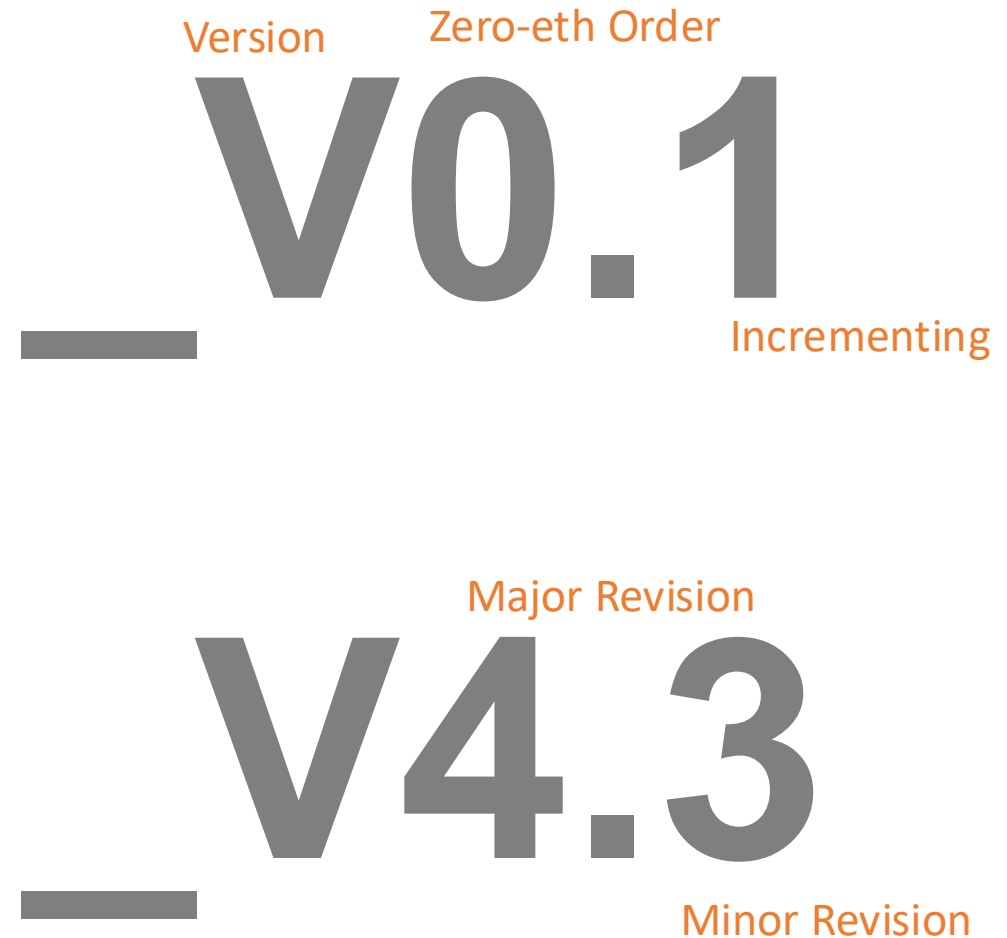
Problem Framing_V0.1.pdf
Problem Framing_V0.2.pdf
Problem Framing_V0.3.pdf
Problem Framing_V0.4.pdf

Primary Persona_V0.1.pdf
Primary Persona_V1.0.pdf

Secondary Persona_V0.1.pdf

Empathy Map_V0.1.pdf
Empathy Map_V0.2.pdf
Empathy Map_V1.0.pdf

Journey Map_V0.1.pdf



Sketching

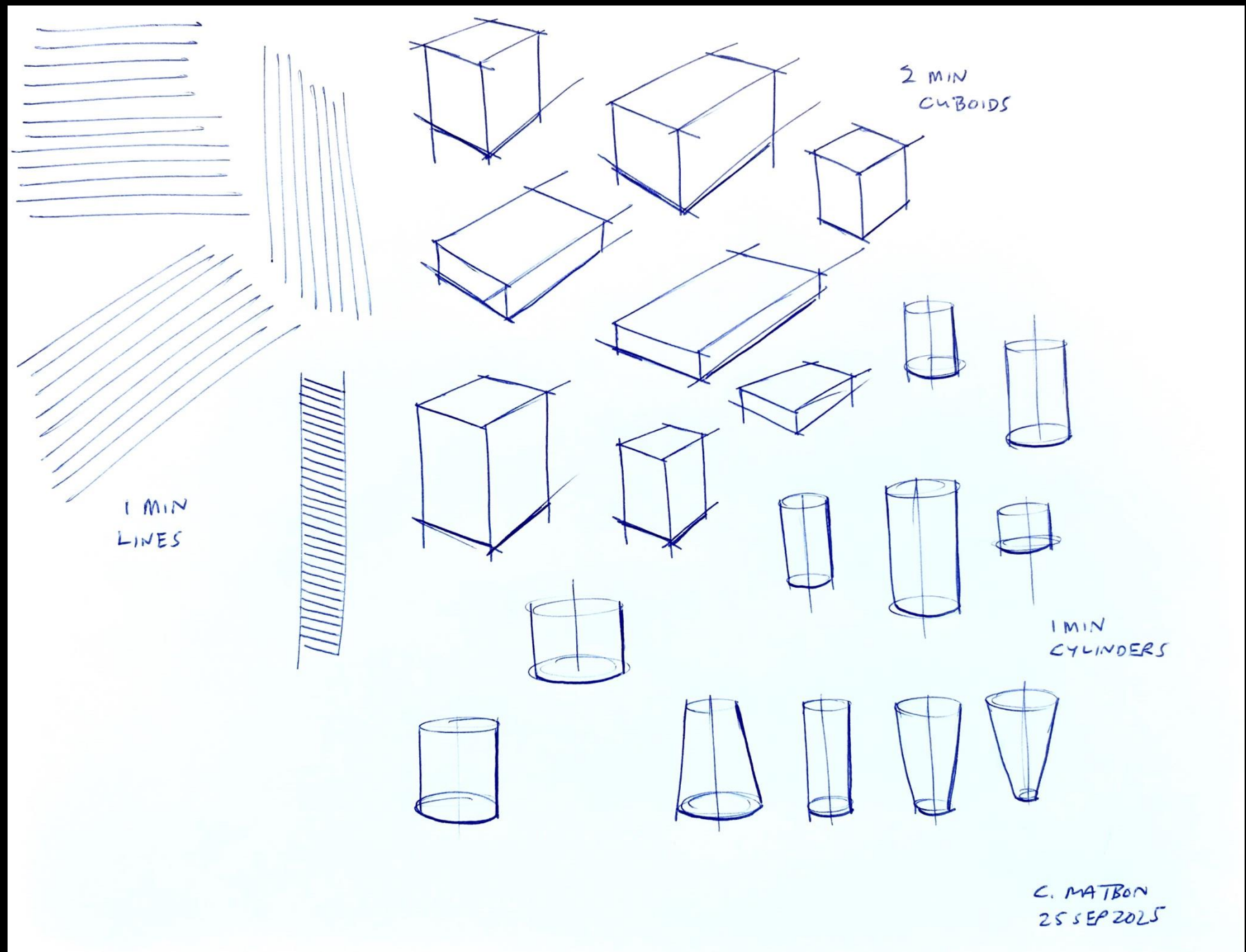
Is everything really just the collection of boxes?

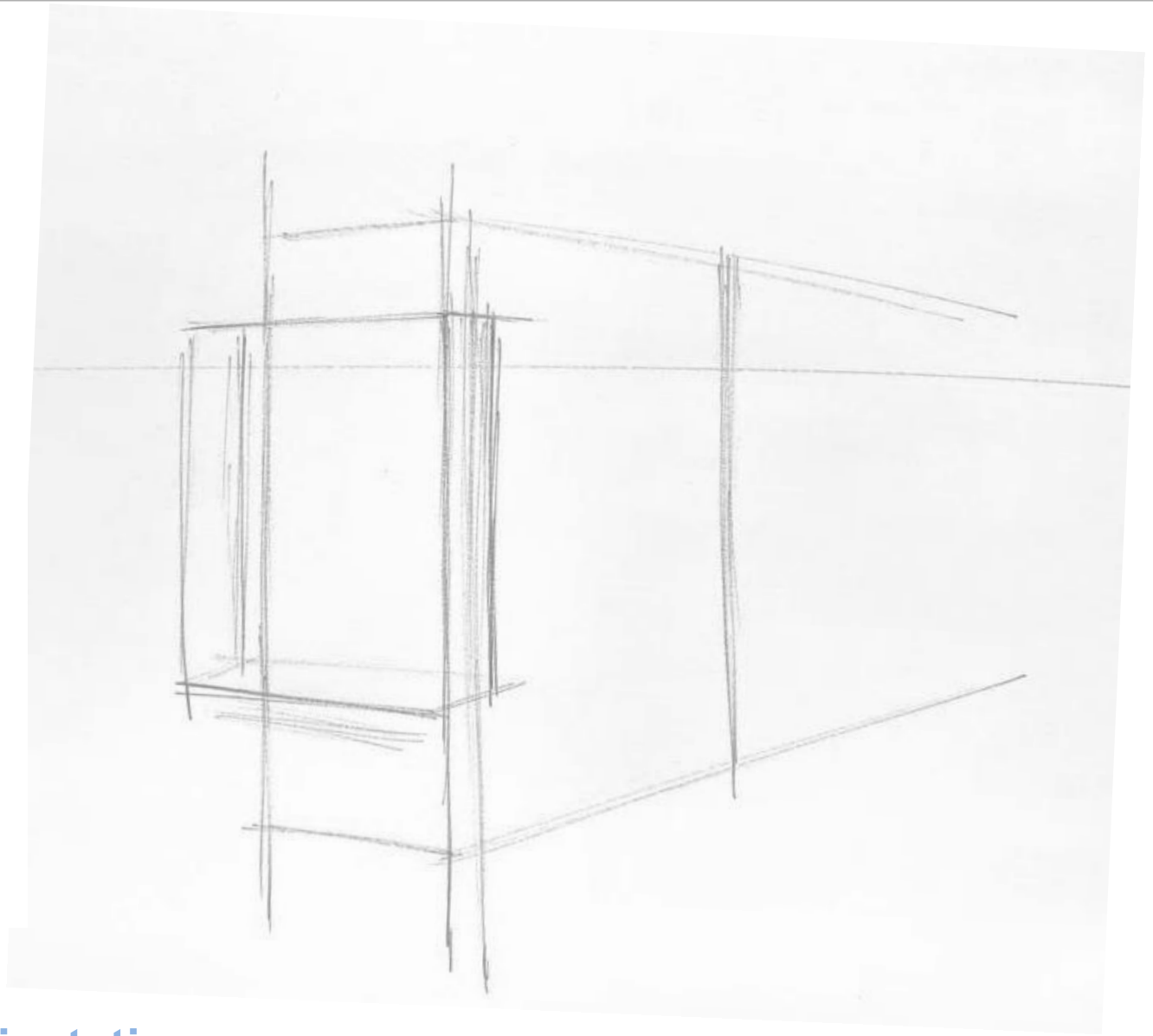
SKETCH warm-up (5 minutes)

1 minute lines

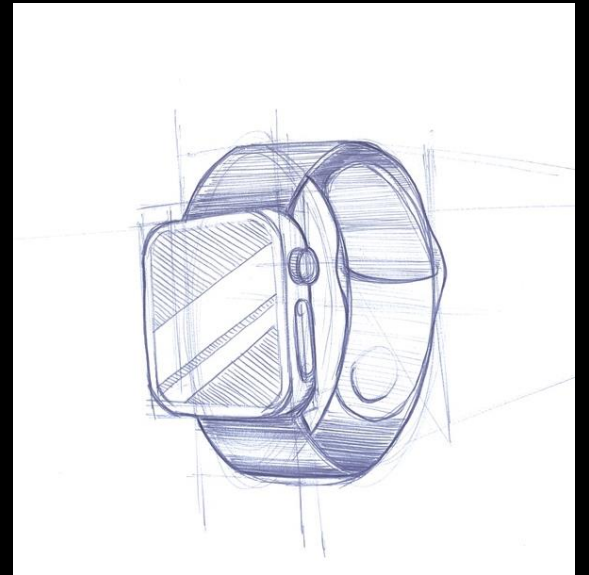
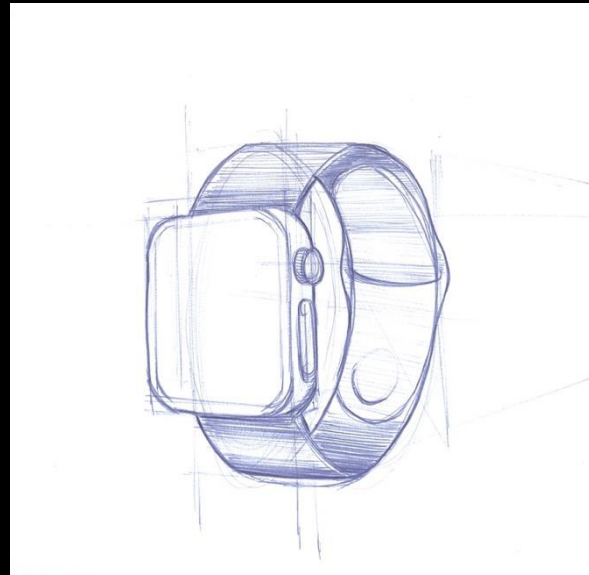
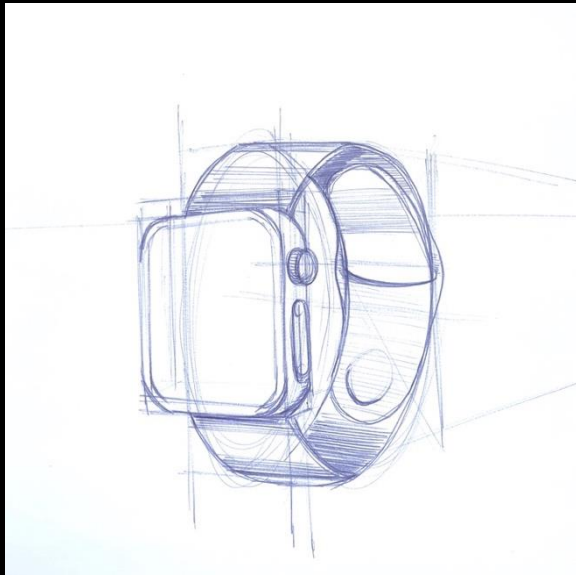
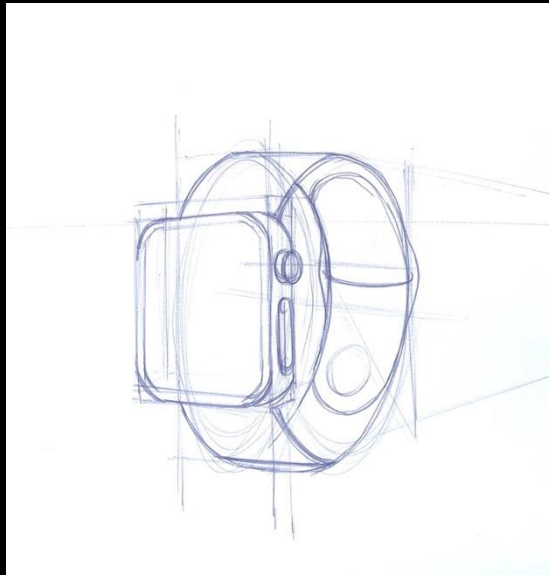
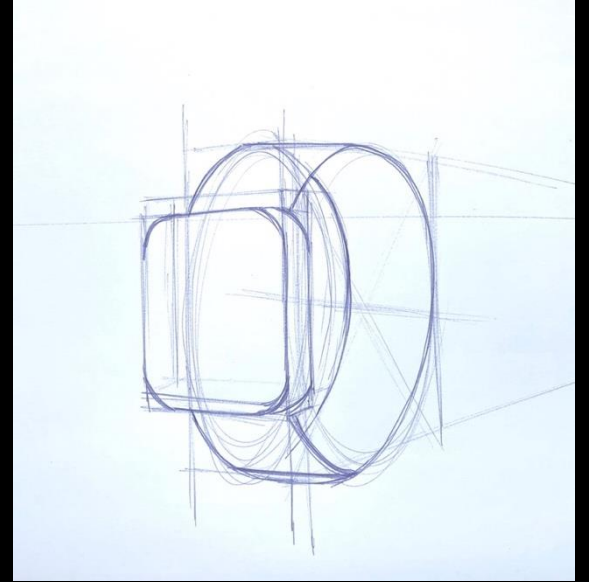
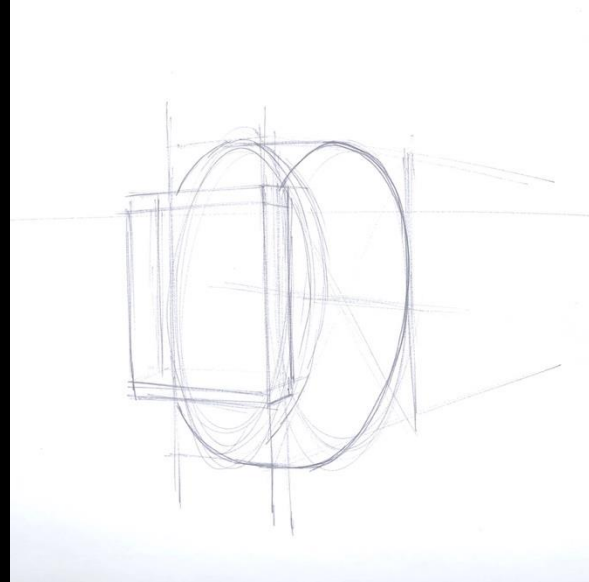
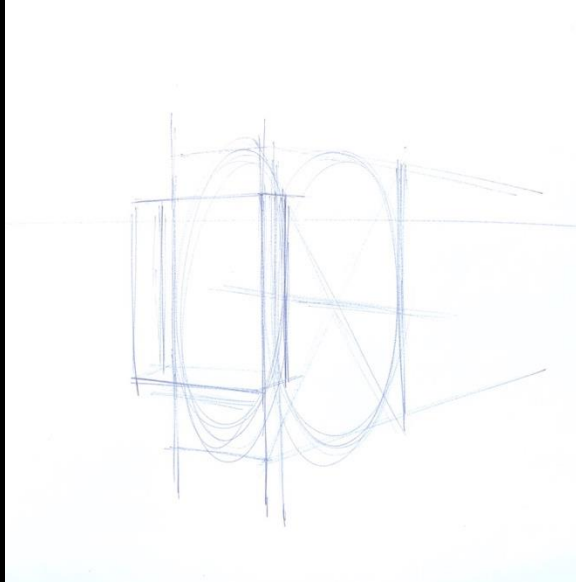
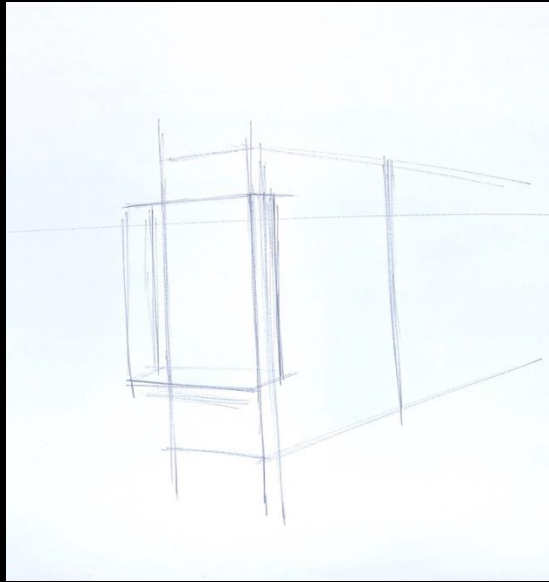
2 minutes cuboids

2 minutes cylinders





**Sketch an Apple Watch in this Orientation.
Try to use construction lines to help.**



Real Innovation Comes from
Deeply Understanding People

What to do now

- Use the Study Guide
- Work on the Project Watch Your Time
 - Preliminary User Insights Packet