EVENT STORMING

Fong Liou

ABOUT ME

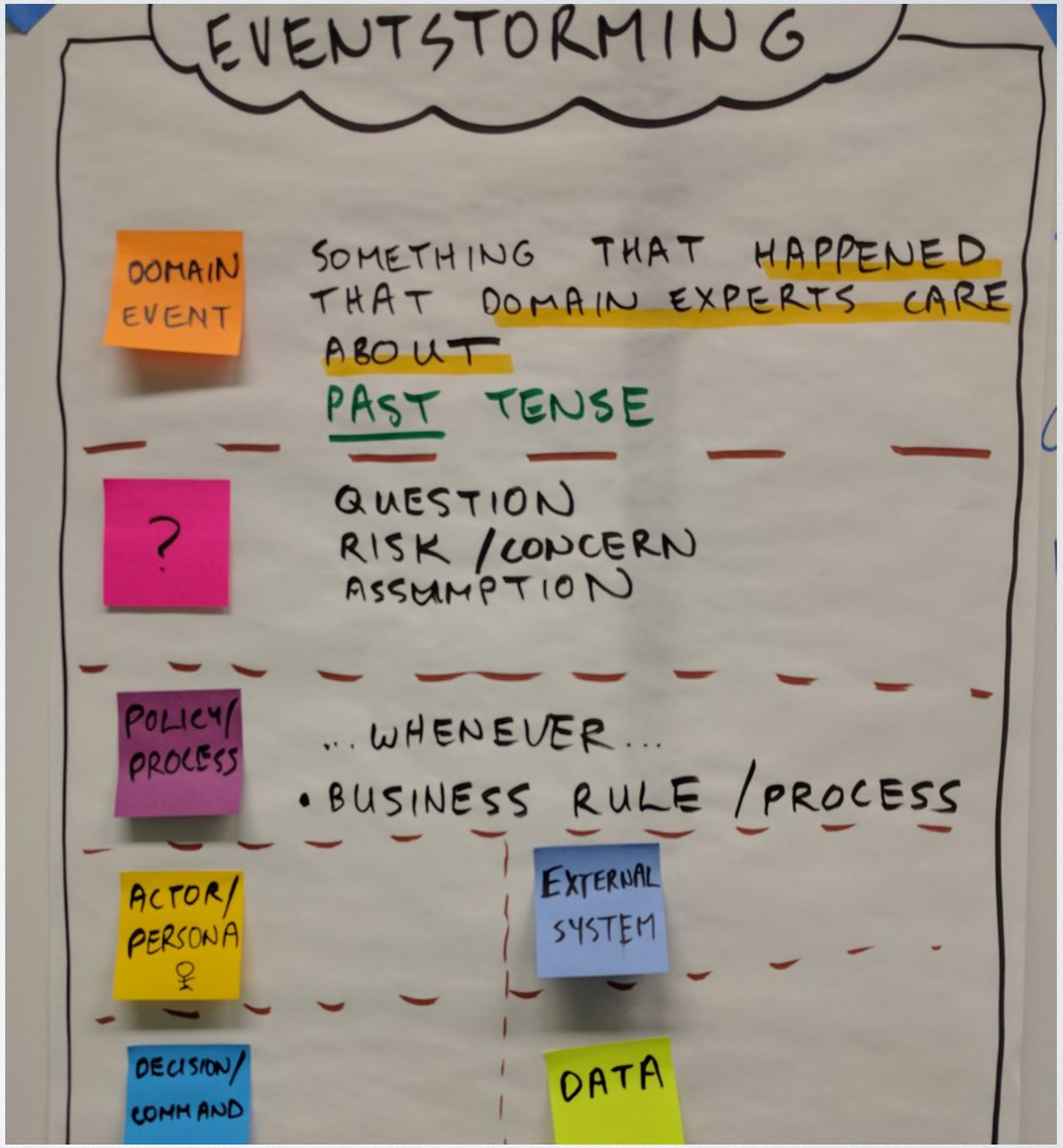
- Fong
- **Backend Engineer**
- **Volunteer, Speaker**
  @Domain-Driven Design TW Community
- **IT 鐵人賽**
  《Think in GraphQL》, 《Think in DDD》

OVERVIEW

- **Intro**
- **Event Storming - Processing Modeling**
  - Events
  - Questions
  - Flesh out
- **Event Storming - Software Design**
  - Aggregate
  - Bounded Context

INTRO

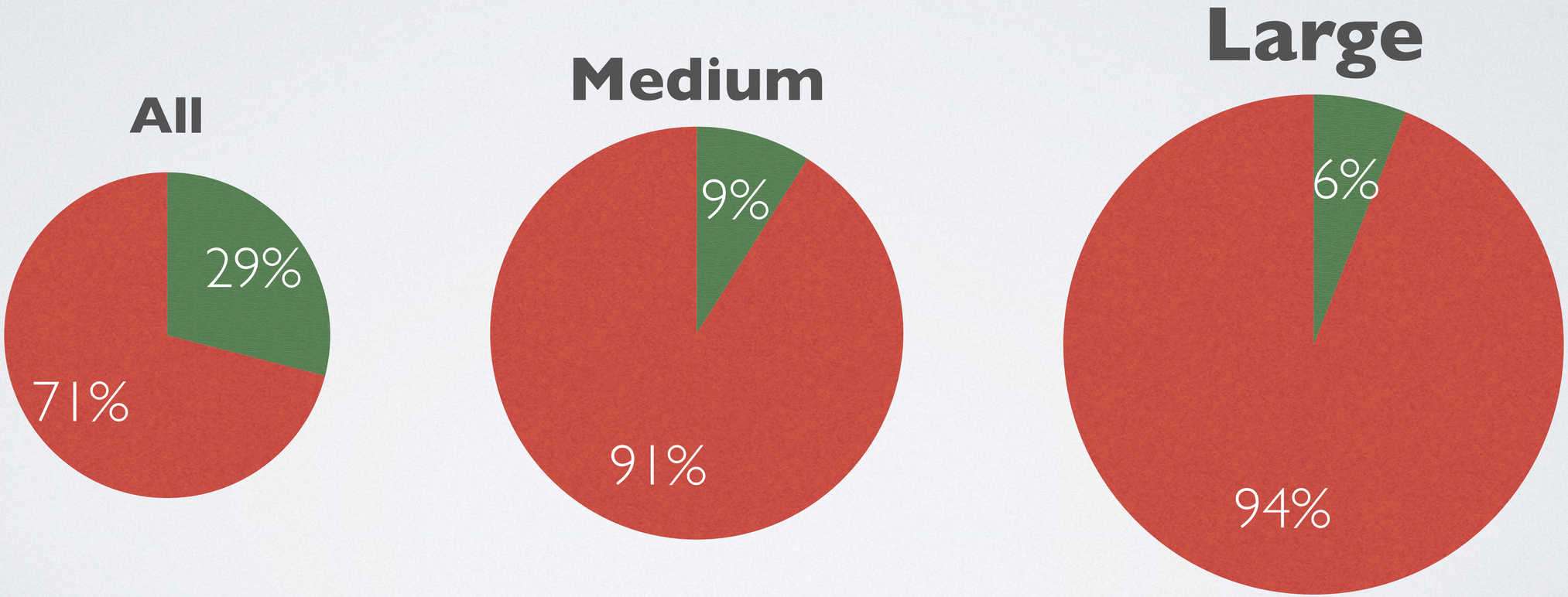
Why Projects Fail



WHY PROJECTS FAIL?

🔥 🔥 🔥

PROJECT SUCCESS RATE



DO YOU LIKE TO READ

SPECIFICATION? 📚 📚 📚

WHY SPEC/DOC SUCKS?

- Insufficient/Changing/Wrong Requirement
- No Error Handling Process
- Messy Legacy Code
- Outdated Documentation

?

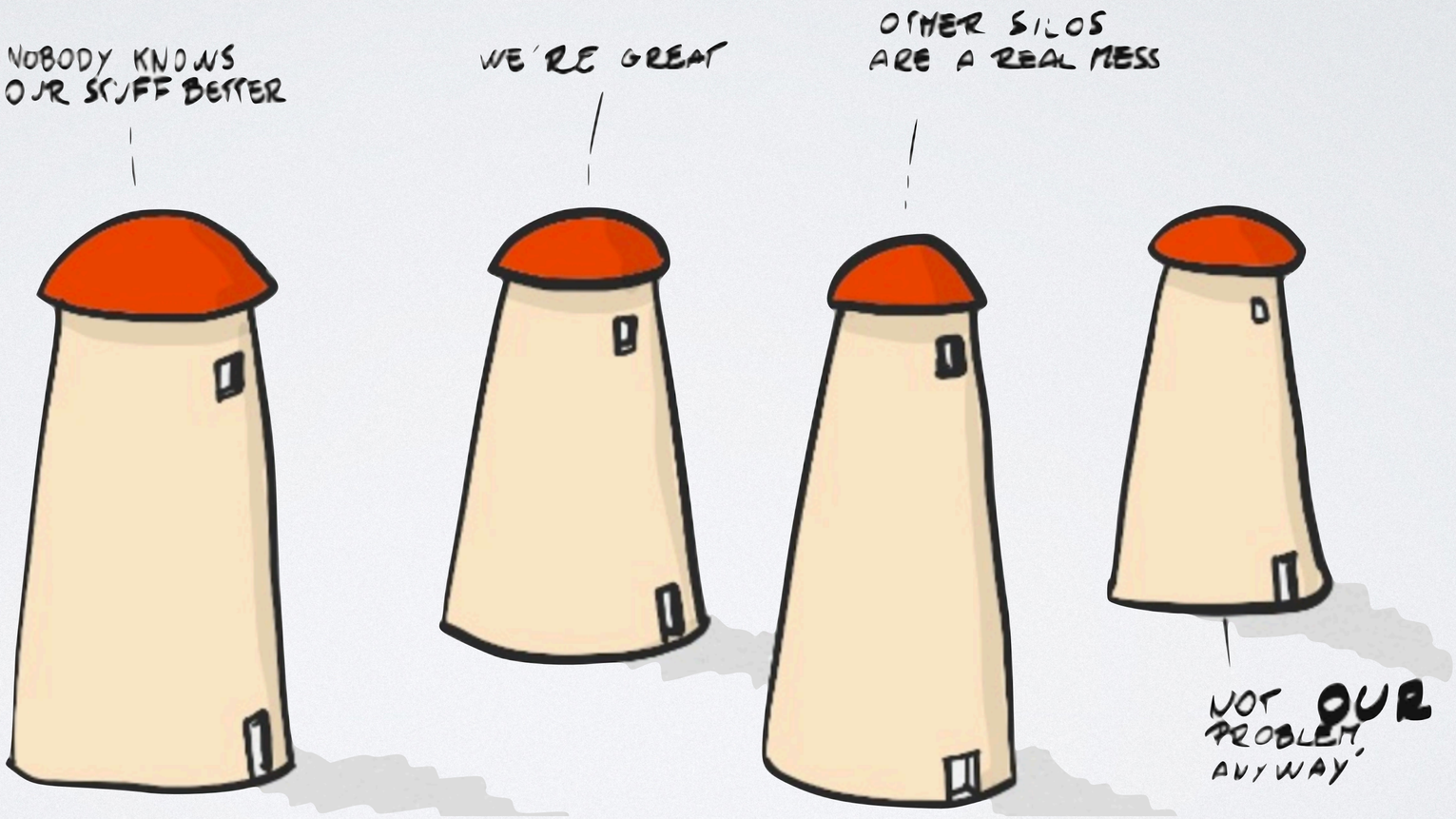
Pokémon

?

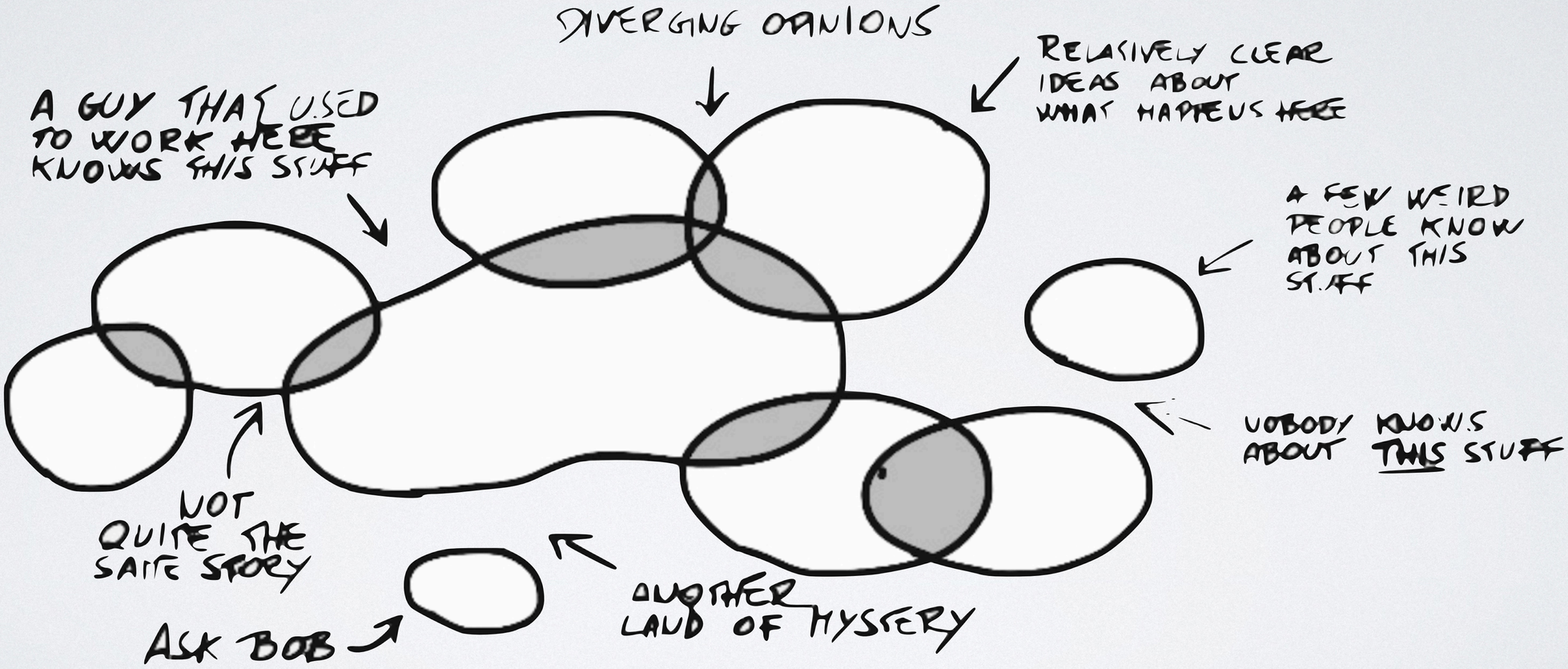
Pokémon

DO THE **RIGHT** THING!

SILOS ARE EVERYWHERE

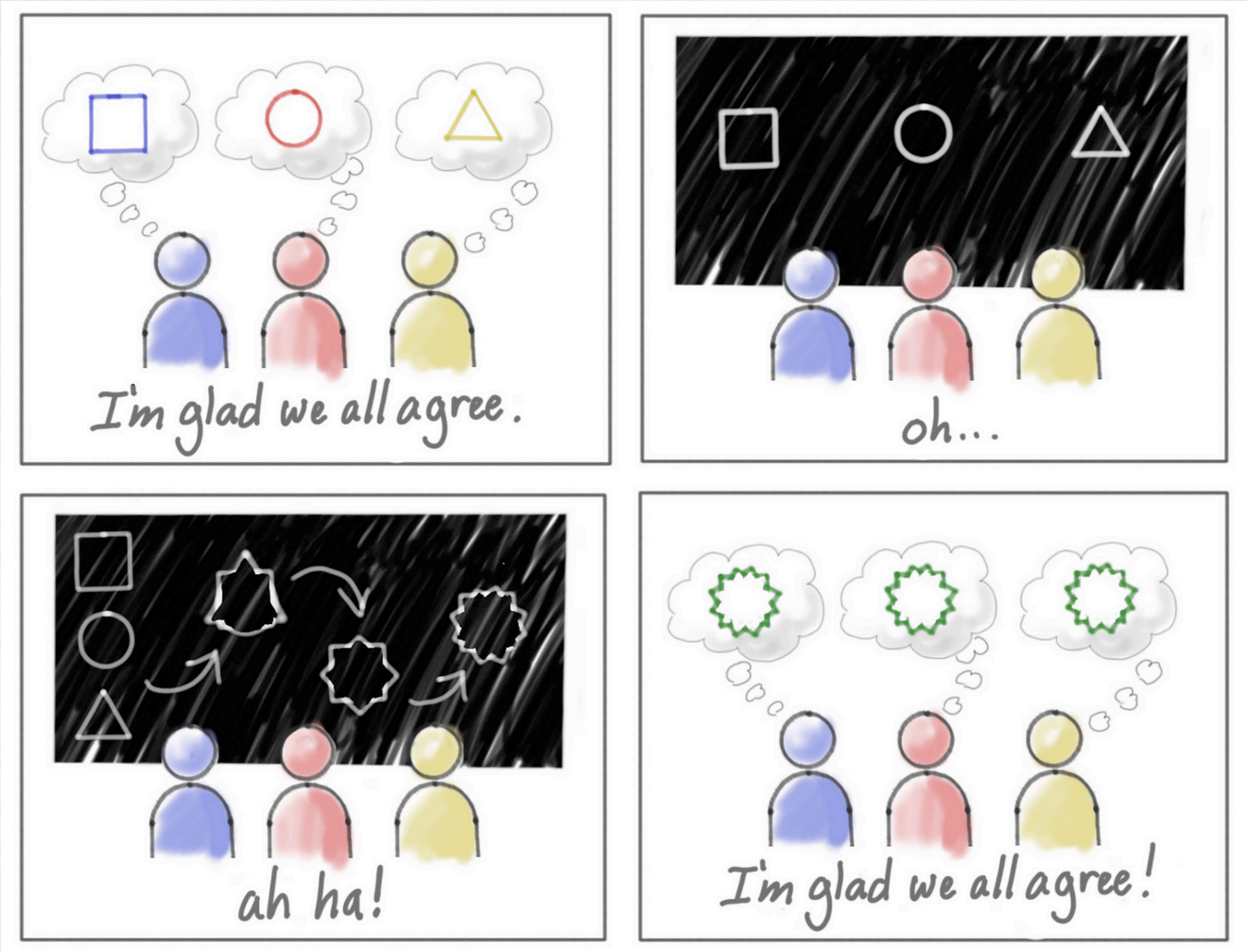


BUT... THINGS ARE NOT EASY



NO ONE CAN GRASP THE
WHOLE PICTURE

COLLABORATION + CONVERSATION = DO THE RIGHT THING



A Fast Way to Learn & Build Consensus… even ***Software Design****!*

EVENT STORMING - PROCESS MODELING

visualize your business process

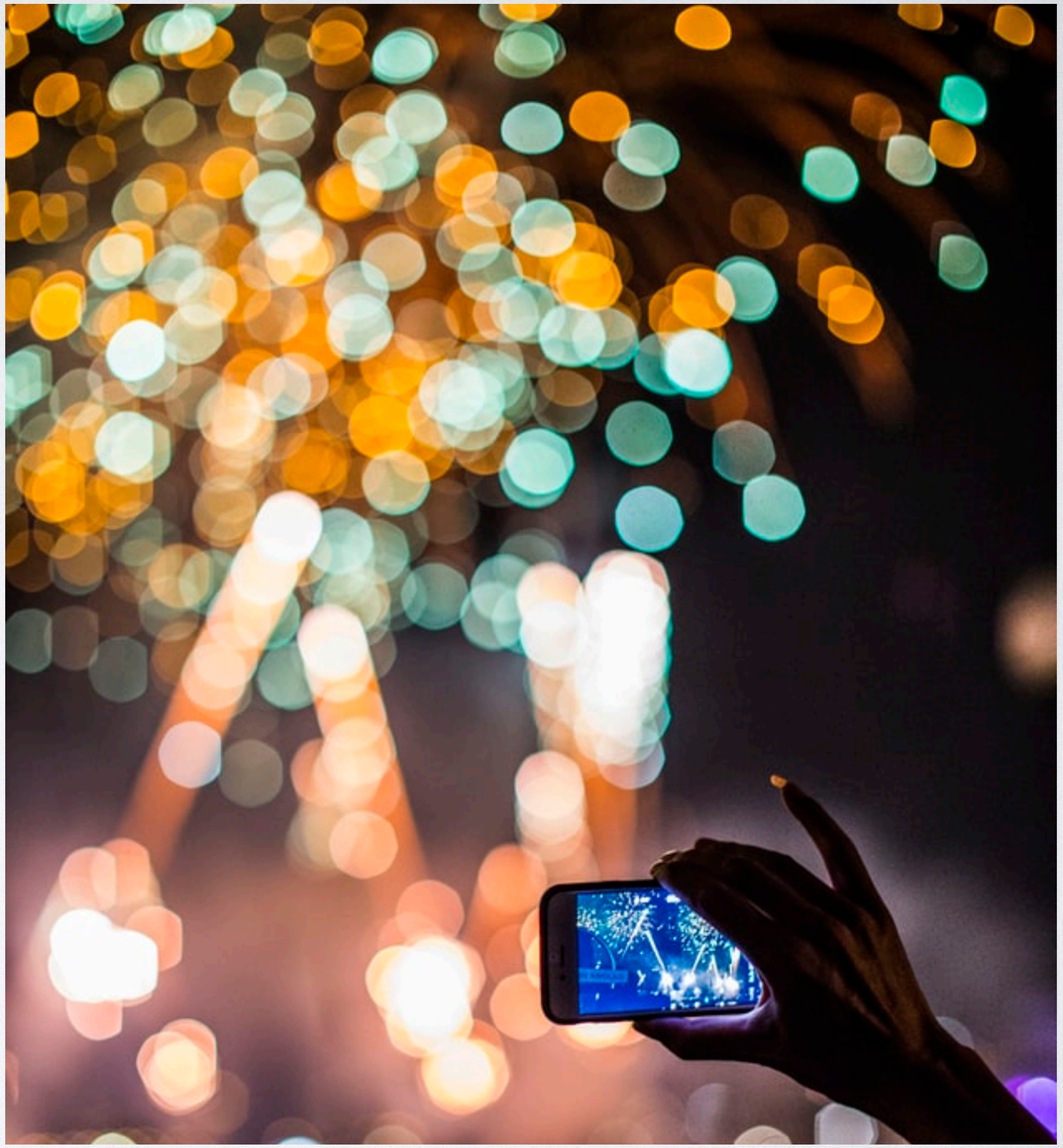


Photo by ben o'bro on Unsplash

DEVELOPING IS A TRIP OF LEARNING

🗺️ **Big Picture With Explicit Process**

🤝 **Collaboration Across Teams**

🔍 **Discover Problems Early**

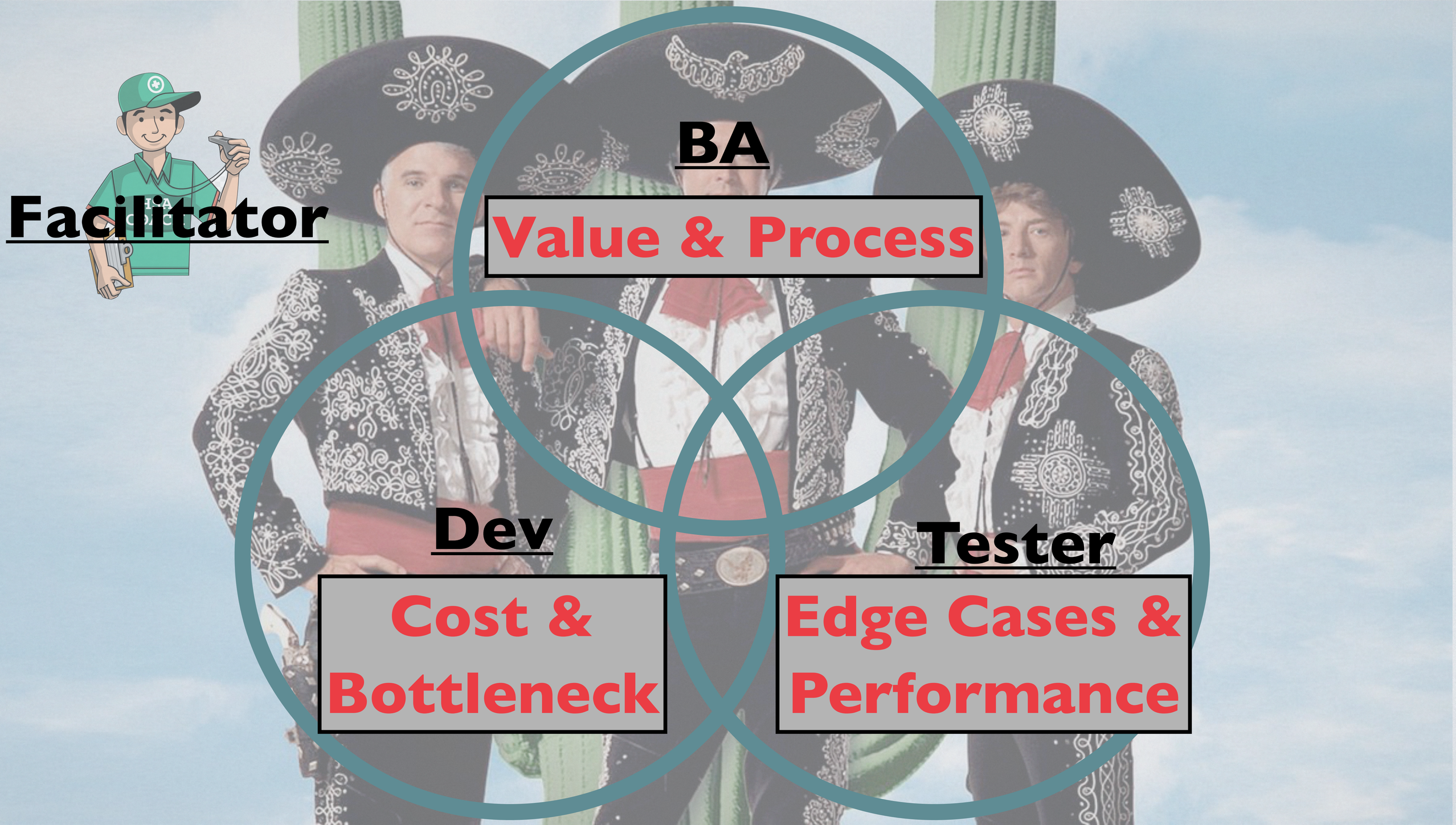
🎯 **Build Consensus**

❌ **Tech Terms**

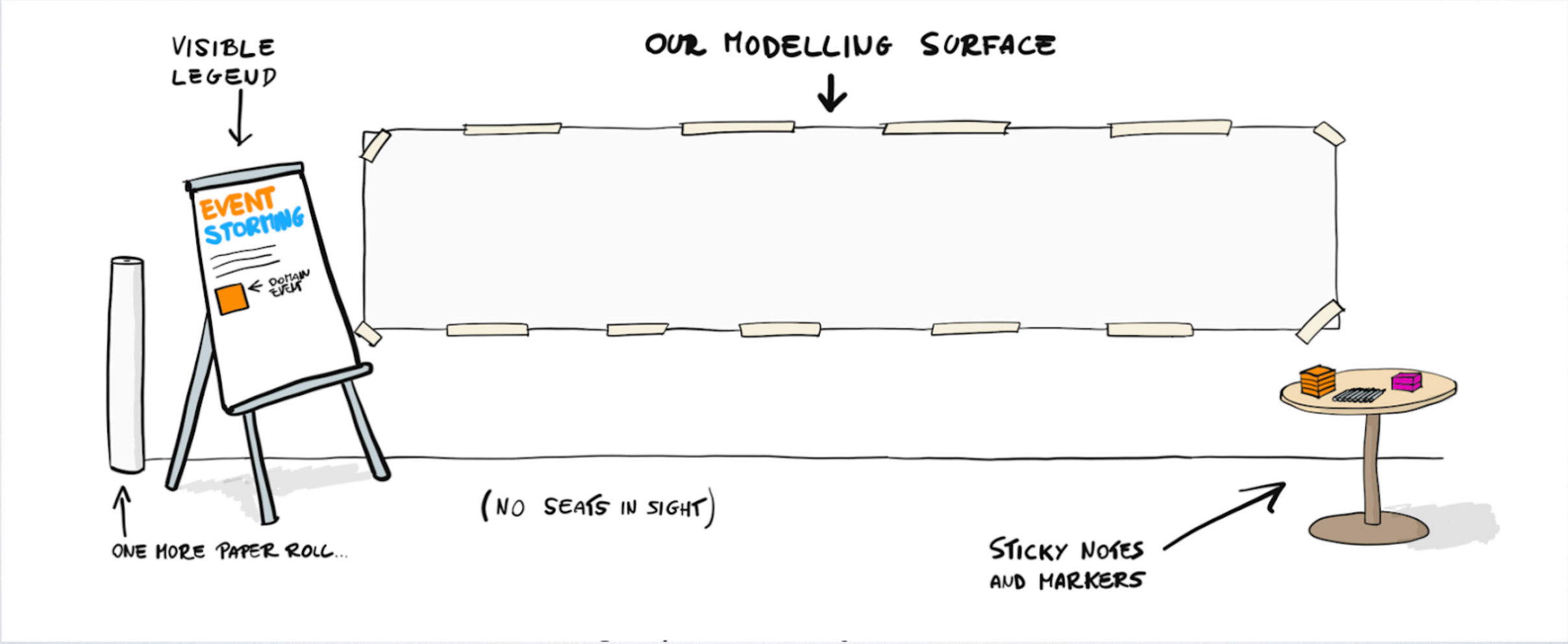
PRE-STEP 1.

FIND THE RIGHT PEOPLE!

BA
(Domain Expert)
Dev
Tester
3 AMIGOS!


Facilitator
BA
Value & Process
Dev
Cost & Bottleneck
Tester
Edge Cases & Performance

PRE STEP 2.
SETUP



...or MIRO.COM

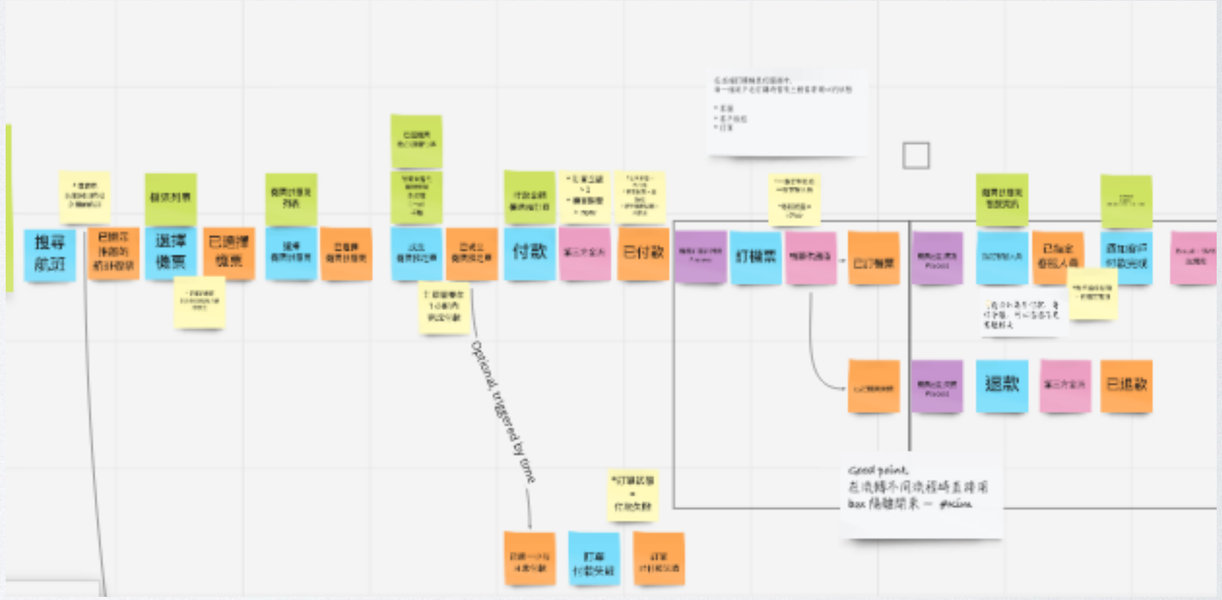
EVENT STORMING TYPES

FROM VISION TO DETAIL (AND BACK)

BIG PICTURE	EVENTS	HOT SPOTS, SYSTEMS, PEOPLE	CONFLICTS, GOALS, BLOCKERS, BOUNDARIES
PROCESS MODELLING	EVENTS	+ POLICIES, COMMANDS, READ MODELS	VALUE PROPOSITION, POLICIES, PERSONAS, INDIVIDUAL GOALS
SOFTWARE DESIGN	EVENTS	+ AGGREGATES	AGGREGATES, POLICIES, READ MODELS, IDS

STEPS OF PROCESSING MODELLING

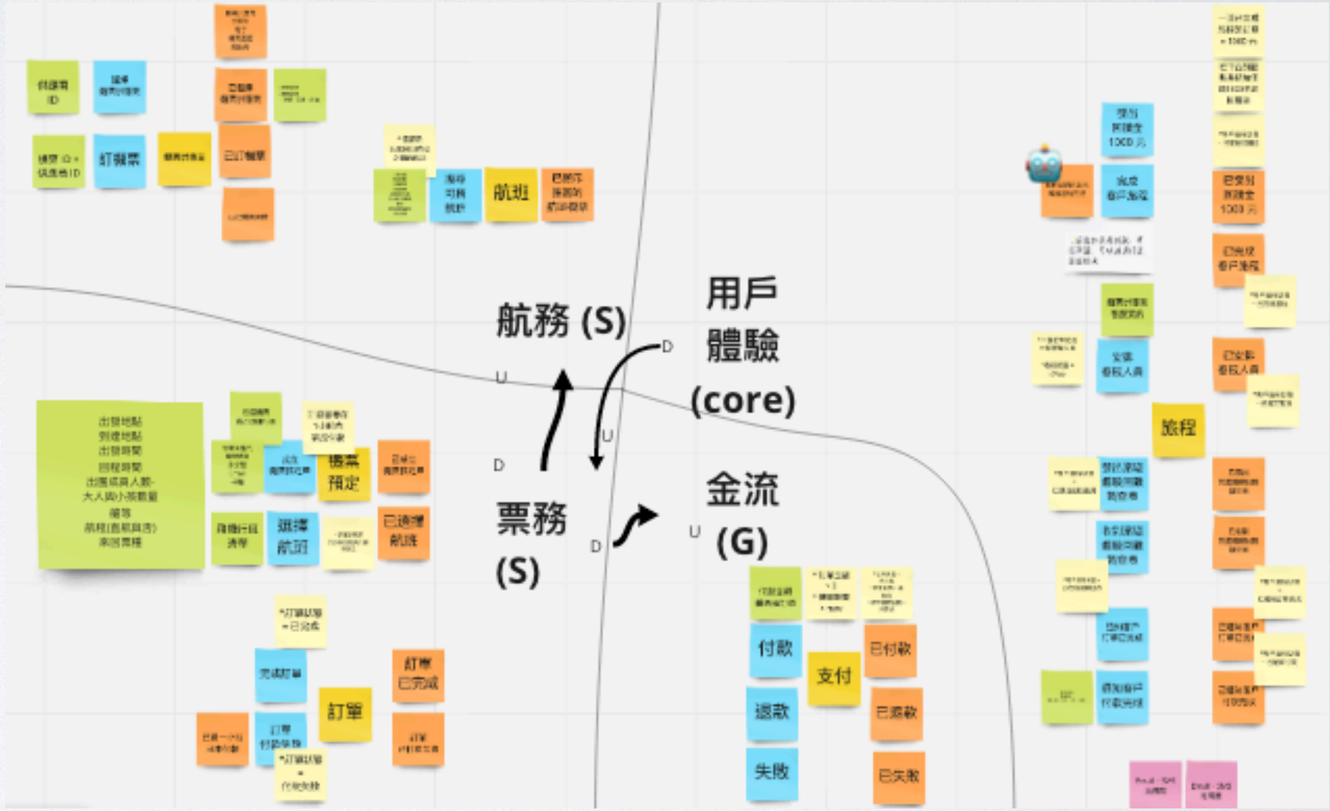
1. Exploring and Organizing **Events**
2. Raise **Questions**
3. **Flesh Out**
    1. Action: Command, Read Model, Rules
    2. Reaction: Process
    3. External System



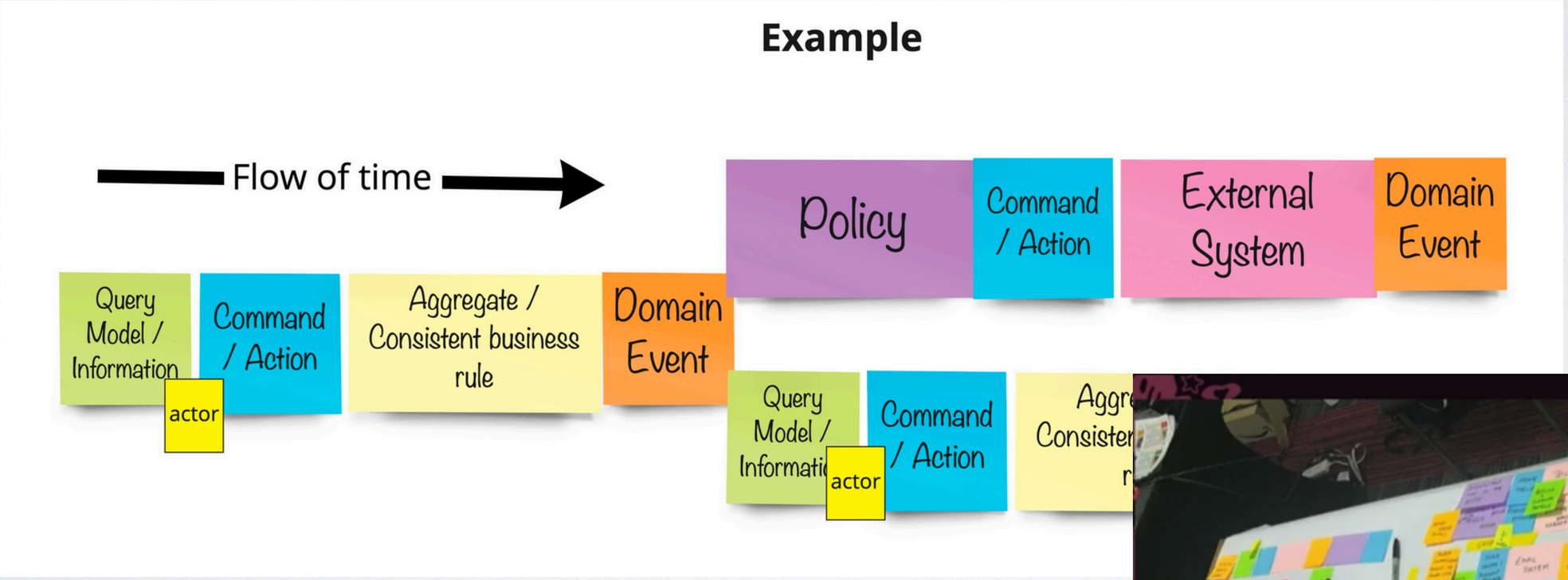
STEPS OF SOFTWARE DESIGN

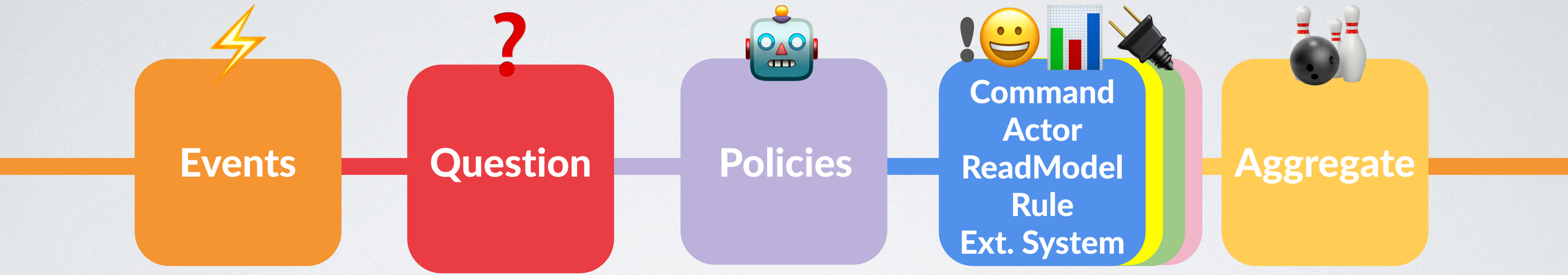
1. Group to Form **Aggregates** (Models in Our Systems)

2. Split into **Bounded Contexts** (Boundary of Our Systems)



EVENT STORMING OUTCOME





1. Explore Events

User Flow of Your Software

2. Raise Questions

Potential Threats of Your Design

3.1 React as Policies

The Automation Part of Your System

3.2 Flesh Out

Details to Design Your System

4. Design Aggregate

Main Characters (Models) of Your System

1. Explore Events

User Flow of Your Software

2. Raise Questions

Potential Threats of Your Design

3.1 React as Policies

The Automation Part of Your System

3.2 Flesh Out

Details to Design Your System

4. Design Aggregate

Main Characters (Models) of Your System

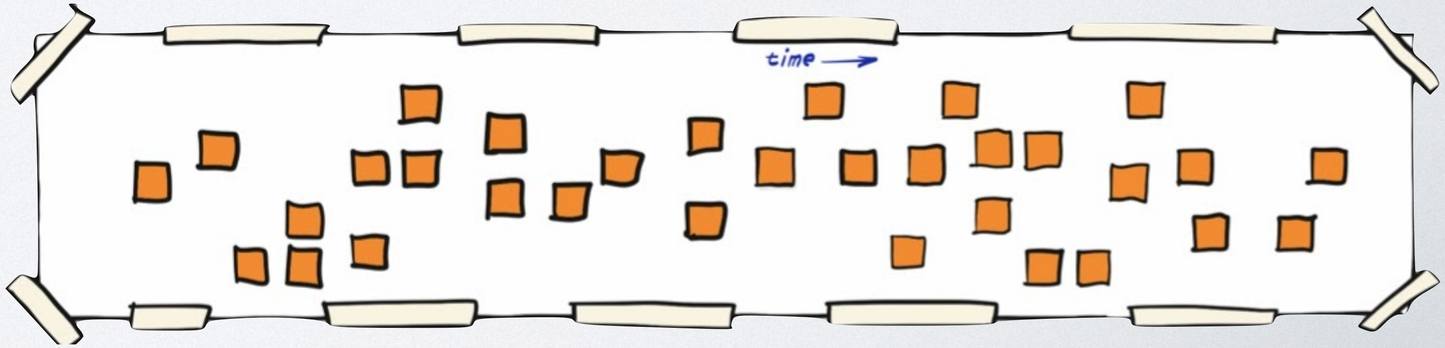
1-1. EXPLORE EVENTS

⚡ EVENT 事件

- Orange
- Something **Already Happened**
  Everybody **Cares**
  (esp. Domain Experts)
- Verb at **Past Tense** (加上已)
- Sorted by **Time:ASC**
- May Have Timing or Reason
  - Capture payments **at** every 3 am
  - Account Locked **for** 3-time login failure

Products Added to Cart

已將商品加入購物車



EVENTS MEANS **EVIDENCE** THAT ALREADY **HAPPENED**

PAY VS. PAID

MOVING VS. ARRIVED

REGISTER VS. REGISTERED

PAY VS. PAID

MOVING VS. ARRIVED

REGISTER VS. REGISTERED

Uncertain State: Implicity

Certain State: Fact

“**Ambiguity** Does Not Compile”

–Alberto Brandolini

EVIDENCE TALKS: LESS AMBIGUITY

BA

我要註冊功能

I want Registration

蛤？

WTF?

已輸入帳密？

Email & Password?

已上傳頭貼？

Avatar Uploaded?

已失敗因為重複帳號？

Duplicate Account?

已發送用戶歡迎信？

Welcome Emails?

已發送新用戶禮券？

Coupons for new members?

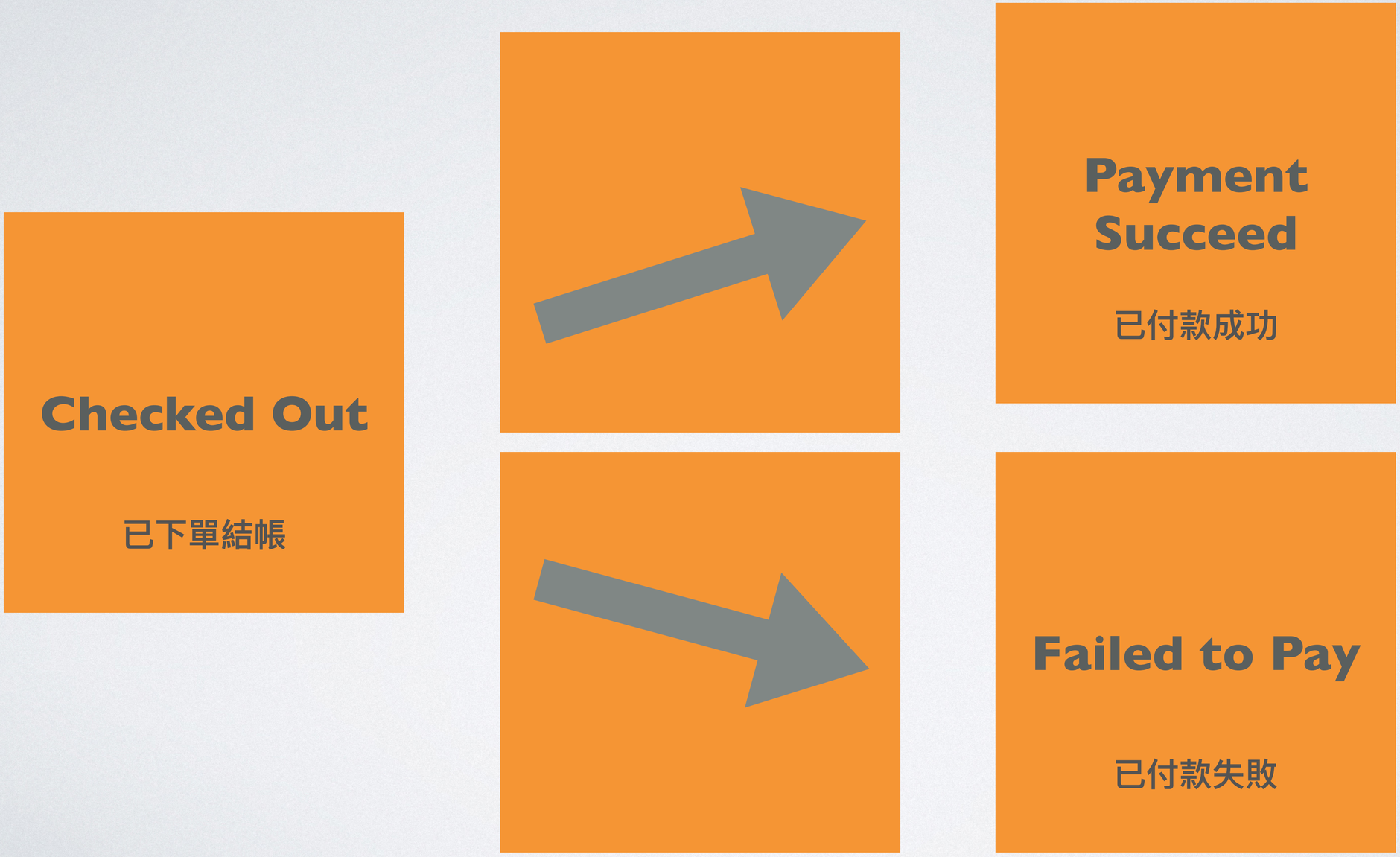
(離線)

(Offline)

Dev

FROM **LEFT** TO **RIGHT**: ENFORCE THE **TIMELINE**

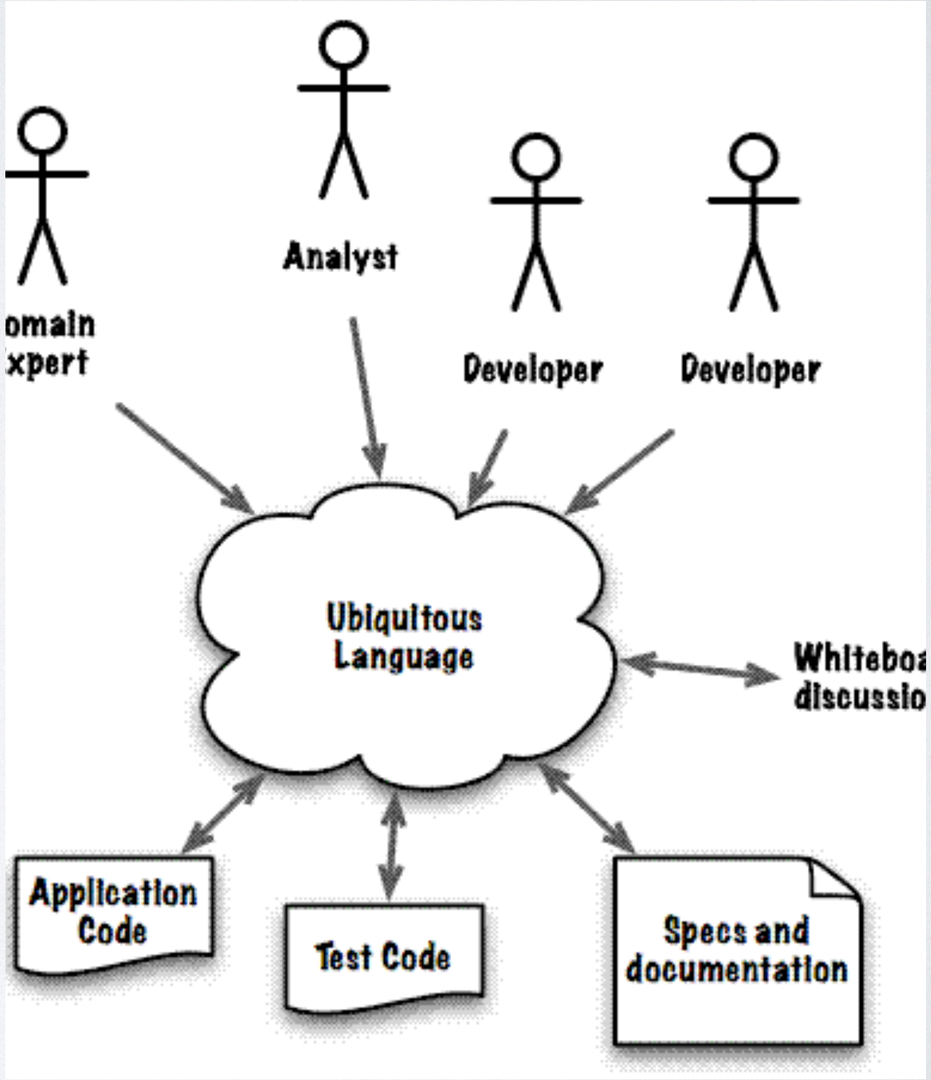
ALTERNATIVE PATHS



Use **Ubiquitous Language**

- Be Careful to the **Terms**
  - Order
  - Bill
  - Ticket
  - Checkout
- Be Critical of the Use of Jargon
- If Different Terms are Used, Maybe They Belong to Different Modules or Systems.

  Then Jot Them down ✏️



SAY NO TO '**CRUD**ISH'

Order Updated

訂單已更新

Order Canceled

訂單已取消

Order Closed

訂單已結案

Order Shipped

訂單已出貨

CASE STUDY - TRIP SERVICE

- 線上一般的出遊服務都僅作為 broker ，幫忙導流量而已
- 用戶體驗不佳，必須自己處理所有的旅程事件
- 關注整體流程 ，讓旅客在旅程中能有專屬的服務人員溝通書裡各種問題

商務場景

- Trip Service 是一家專營完整出遊服務的公司，目標是提供一站式的機票、酒店、汽車出租服務。
- 當前首要的業務範疇以機票服務為主，旅客只需在此平台下訂機票，便能完成付款與取得正式電子機票憑證作業。
- 目前正在思索該如何建構這樣的平台來支撐業務

[PRACTICE]
EXPLORE YOUR EVENTS IN
20 MINS

請各組推派兩名 *Domain Expert* 到前面來領取情境敘述

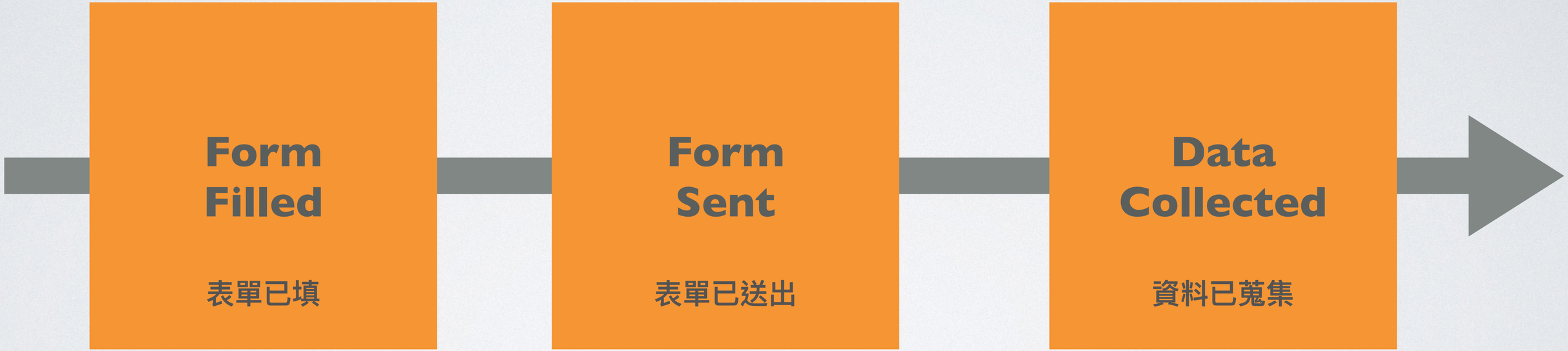
Domain Expert Puts Down **the First One**

[PRACTICE]
EXPLORE YOUR EVENTS IN 20 MINS

- 多**與 Dmomain Experts 討論**，請減少私下交談
- 挖出**細節**，不要太抽象。盡可能找出更多事件
- 先以 **Happy Path** 為主
- 每個人都可以貼，重複也沒關係

1-2. **ORGANIZE EVENTS**

Ask Again: Do We Really Care About?



[PRACTICE]
ORGANIZE EVENTS IN
15 MINS

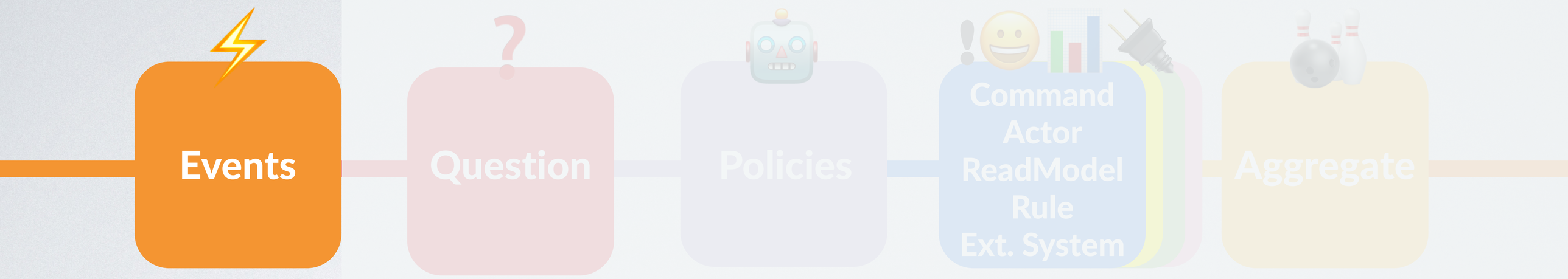
[PRACTICE] ORGANIZE EVENTS IN 15 MINS

- 選出一個人用說故事模式，從頭順到尾。
- 小心 CRUDish Style & 注意 Events 順序性
- 把重複的拿掉或是把概念相近的合併
- 把錯誤格式改正 (ex: 下訂單 ➡ 已下訂單)
- 當使用不同 Term 描述時，先討論後再修正

QUICK QUESTIONS 🤔 - 5 MINS

- 使用 Event 來溝通跟過去的方式有什麼不一樣的地方？
- 使用過去式來表達有遇到什麼困難？
- 看到整個流程後有什麼想法？

WHAT IF WE HAVE QUESTIONS ABOUT THE EVENTS?



1. Explore Events

User Flow of Your Software

2. Raise Questions

Potential Threats of Your Design

3.1 React as Policies

The Automation Part of Your System

3.2 Flesh Out

Details to Design Your System

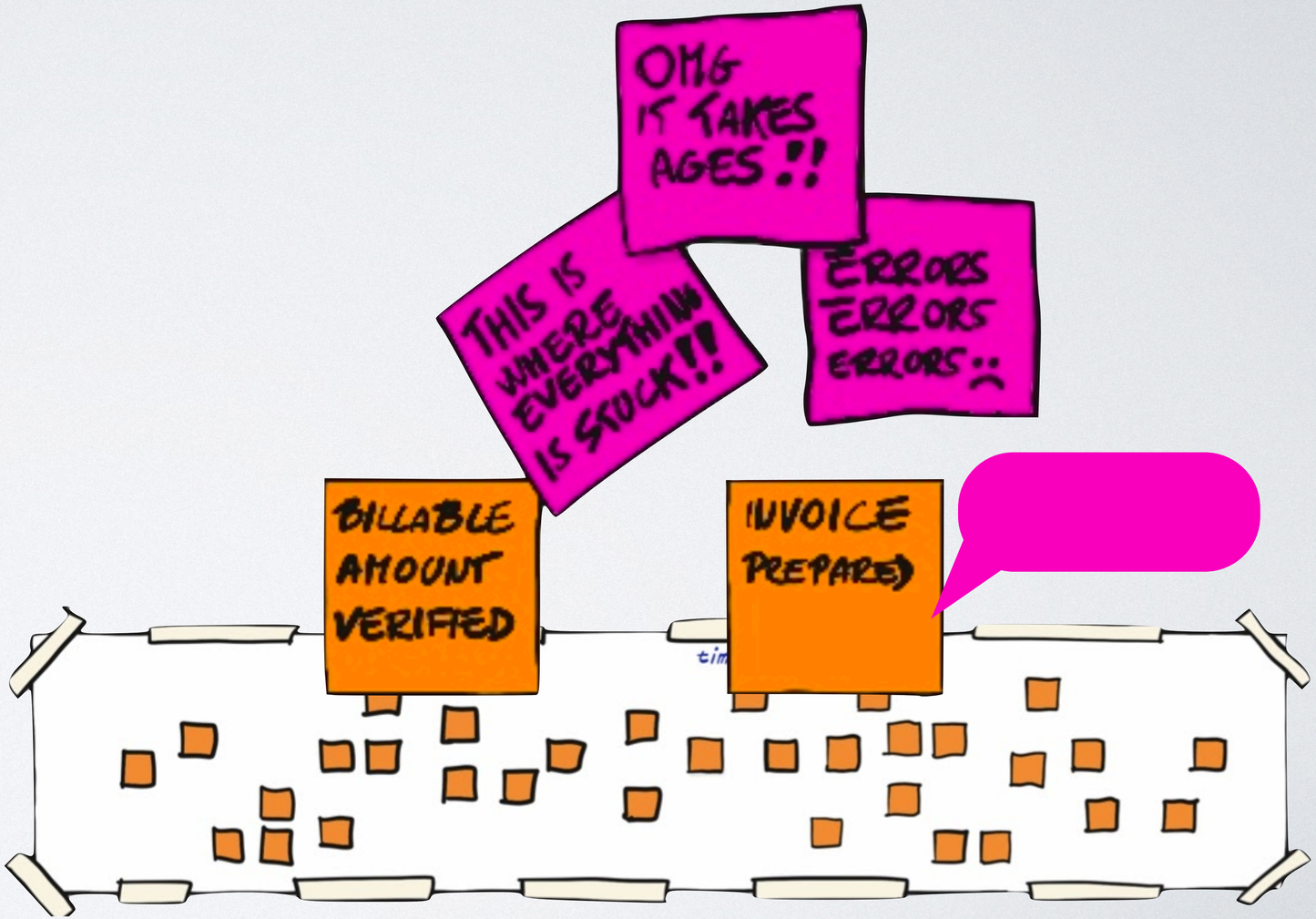
4. Design Aggregate

Main Characters (Models) of Your System

2. **RAISE QUESTIONS**

? QUESTIONS 問題

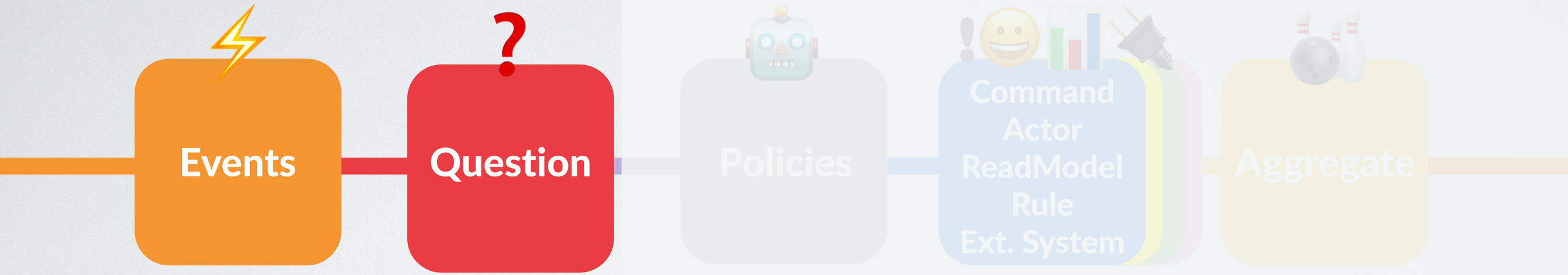
- Red
- Rotate to **45°**
- When to Use?
  - **I don't Understand**
  - **Something Wrong**
  - **Too Hard to Implement**
- **Put off** the Controversy
- Point out the Sad Path



[PRACTICE]
2. QUESTIONS STORMING
15 MINS

[PRACTICE]
2. QUESTIONS STORMING
15 MINS

- 前 5 mins: 貼上問題；後 5 mins 回答問題
- 把有問題的地方標注上 Question
- 如果該 Question 可以被解答/解決，就在旁邊註記上去或作相應調整
- 如果無法立即解決，就先記錄下來並擱著



1. Explore Events

User Flow of Your Software

2. Raise Questions

Potential Threats of Your Design

3.1 React as Policies

The Automation Part of Your System

3.2 Flesh Out

Details to Design Your System

4. Design Aggregate

Main Characters (Models) of Your System

3-1. **REACT AS A POLICY**

It is not the **domain experts knowledge** that goes to production, it is the **assumption** of the **developers** that goes to **production**

– Alberto Brandolini

DOES YOUR KNOWLEDGE EQUAL MY **ASSUMPTION**?

WHAT WILL HAPPEN WHEN

NEW USERS REGISTERED?

WHO LIES?

BA

為什麼新會員沒有拿到禮物？

Why new members doesn't get gifts?

用戶**一旦**註冊後系統**就會**先寄 Email 然後發購物金 100 元吧？

Once registered, it'll send email then give $1000

錯！新會員政策應該是「註冊就送 S 級火槍兵」

Wrong! Whenever registered, they get s-level musketeer

(離線)

(Offline)

Dev

User Registered

用戶已註冊

Welcome Email Sent

已寄送歡迎信

Reward Points Sent

已寄送購物金

S level Musketeer Sent

s 級火槍兵已送出

User Registere
用戶已註冊
New User Policy
新用戶政策
elcome ail Sent
寄送歡迎信
Reward Points Sent
已寄送購物金
S level Musketeer Sent
S 級火槍兵已送出

User Registered

用戶已註冊

New User Policy

新用戶政策

Welcome Email Sent

寄送歡迎信

Reward Points Sent

已寄送購物金

S level Musketeer Sent

S 級火槍兵已送出

ARE YOU SURE **”RIGHT AWAY”** ?

活動名稱 【DDD TW 7月沙龍】用 DDD & BDD

邀請碼管理

名稱	有效期限		數量	狀態
社群之友 - 6 月 meetup 參與者	2020/07/25 23:50 (+0800)	社群之友票（限參與 6 月 meetup 者）	0 / 4	正在產生

+ 新增邀請碼

數量 狀態
0 / 4 正在產生

UBISOFT 取消訂閱

哈囉 fongx777,

系統已確認您的偏好設定。請注意，系統變更需 48 小時才會更新生效，接下來一至二天您可能會繼續收到信件。

，系統變更需 48 小時才會更新生效，

別忘了，您隨時可以回來 http://account.ubisoft.com/ 更新帳號設定、新增或刪除您有興趣的遊戲品牌、更新連結帳號，還能取得 Ubisoft 全套完整服務。

還有其他問題或建議嗎？歡迎至 https://support.ubi.com/ 與我們聯絡。

Order Delivered

訂單已出貨

Order Paid

訂單已付款

30 mins Later &Pizza not Delivered

超過 30 mins 未送達

30-minute Delivery Guarantee Policy

外送限時送達政策

Order Delivery Timed Out

訂單金額已免費

Drivers Penalized

送貨員已懲罰

🤖 POLICY (流程/業務政策)

- How is **Our System Supposed to React** to **Given Events**?
- Whenever **Events**, Then **Command**
- Most Misleading Part Between Techies & Non-Techies
- When to Use?
  - 當聽到「**一旦...就...**」、「**立即**」
  - 當你在過程中遇到一段**複雜流程**時，可以先貼上 Policy

- Involves Automation, Non-automation Processes
  - 自動化（轉帳成功後需要自動寄簡訊）
  - 非自動（服務提供商提供報價後需要跑內部簽核流程）

Order Checked Out

訂單已結帳

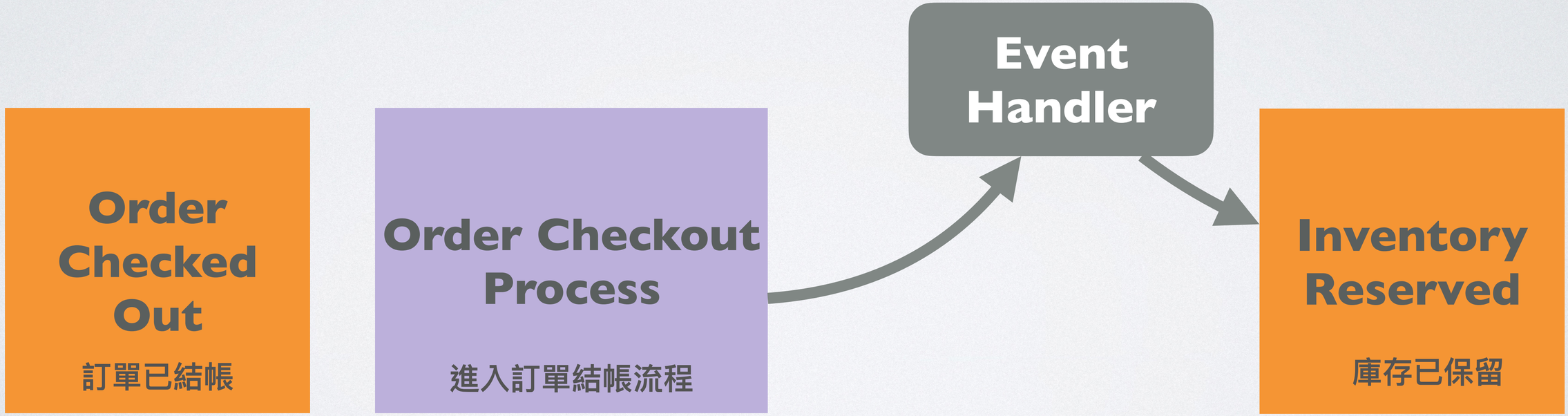
Order Checkout Process

進入訂單結帳流程

Inventory Reserved

庫存已保留

- 對於工程師來說：對於系統實作有更深的認識，把 Process 當作 Event Handler 機制
- 對於領域專家來說：更理解系統的流程與運作



[PRACTICE]
REACT AS POLICIES IN 5 MINS

[PRACTICE] REACT AS POLICIES IN 5 MINS

- 當你在過程中遇到一段**複雜流程**時，可以先貼上去 Policy
- 當聽到**「一旦...就...」**、**「立即」**就貼上或換成 Policy
- 一個 Policy 可能會觸發一連串的 Event，可以標記出開始與結尾

MORE EXAMPLES

購物車已結帳

結帳流程

已保留庫存

已建立訂單

已付款

已送出報價

人工簽核流程

簽核成功

簽核失敗

已申請司機

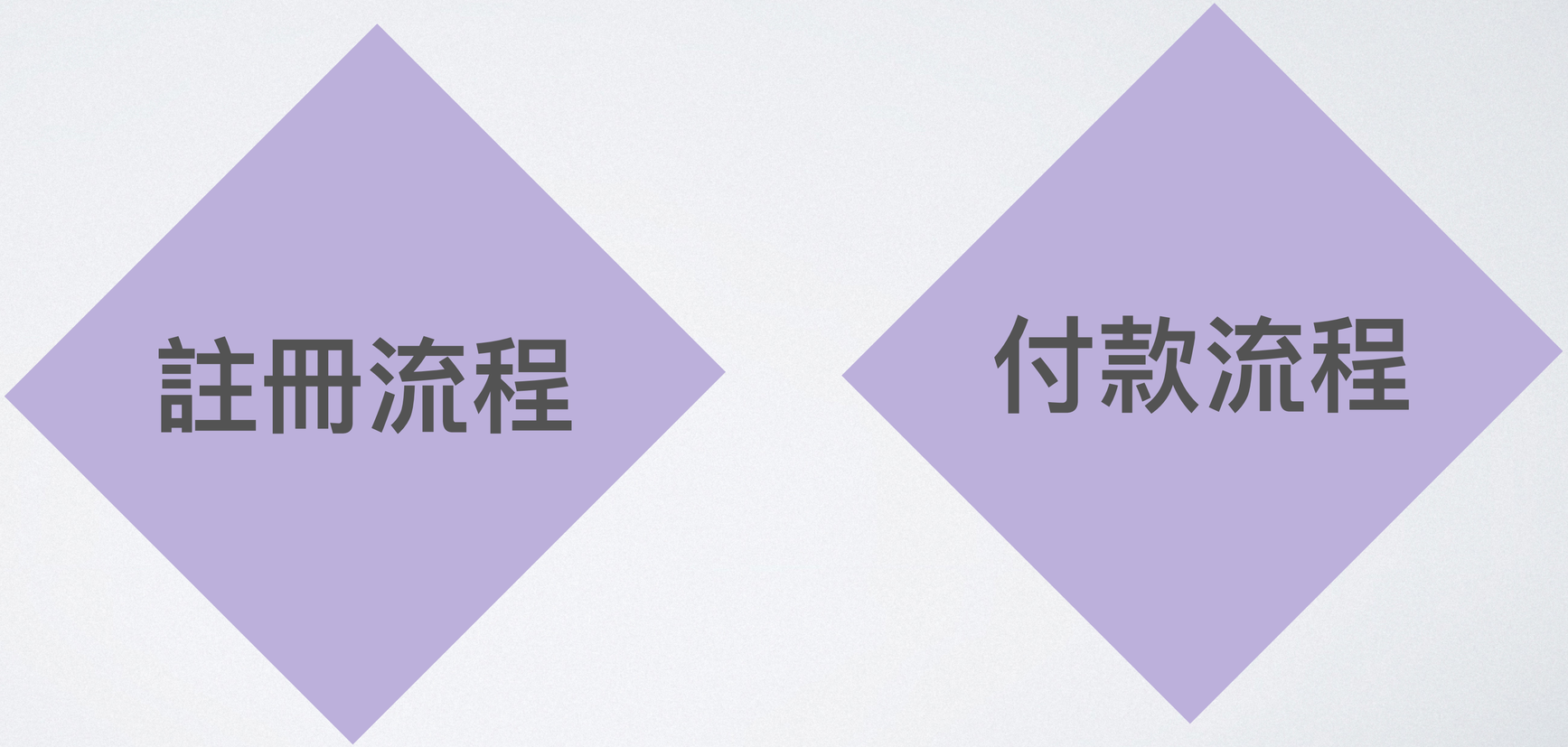
派車演算法

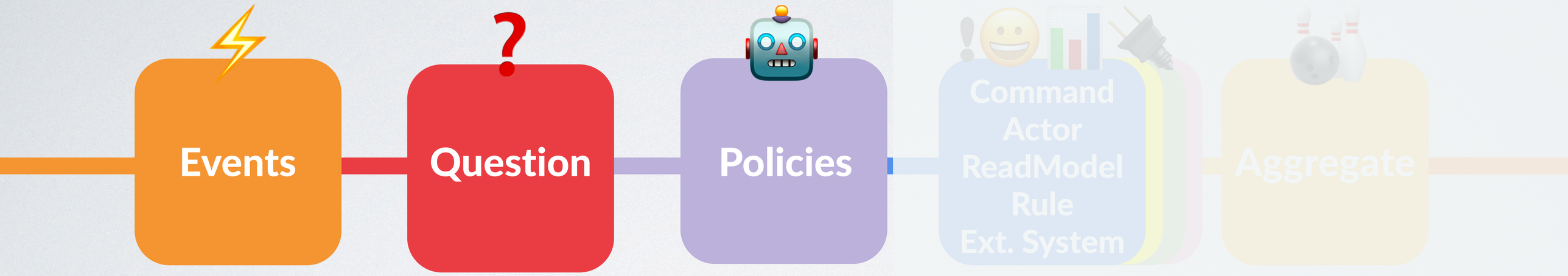
已叫車

HOW ABOUT A **COMPLEX** PROCESS WE DON'T HANDLE NOW?

進階用法：PROCESS

- 當我們不想要深入探討一些細節時，可以用 Process 來代表複雜流程





1. Explore Events

User Flow of Your Software

2. Raise Questions

Potential Threats of Your Design

3.1 React as Policies

The Automation Part of Your System

3.2 Flesh Out

Details to Design Your System

4. Design Aggregate

Main Characters (Models) of Your System

3-2. **FLESH OUT**

AFTER THE WHOLE FLOW, LET US FLESH OUT ‘ACTUAL BEHAVIORS’ (補上真實行為)

! COMMAND 命令 / 😀 ACTOR 使用者

- Blue
- **Decisions made by Users/Softwares**
- Trigger Events
- Usually Related to Events
- In Implementation, They might be an API or a Class Method

- Yellow, Smaller Rectangle
- User or Software

READ MODEL 讀模型

- Green
- **Information** that Actor Needs to Make the Command (Decision)
- EX
  - 「顧客」在「簽約」時要先「詳閱公開說明書」
  - 「顧客」要「把商品加入購物車時」要先「知道價格」

$ 3.1 Bil.

總金額高達
31 億新台幣！！

Customer

顧客

Buy Lottery Tickets

買威力彩

[PRACTICE]
FLESH OUT
10 MINS

[PRACTICE] FLESH OUT 10 MINS

- 為 Event 加上對應的 Command (已下訂單 vs 下訂單)
- 為這個 Command 尋找 Read Model
- 思考：這個 Command 跟這個 Event 是否相搭？是否在乎該 Event
  (已輸入　已搜尋　已顯示 → 搜尋　已顯示)

✔ RULE

- Rules Need to be **Followed** between Command and Events
- **Business Rules/Logics** of the System
- Invariants

Rule

Check out Order

結帳訂單

Max 8 Items

最多 8 個品項

Order Checked Out

訂單已結帳

Build Car

打造車子

4 Tires Required

要有四個輪胎

Car Built

車子已打造

Transfer Money

匯款

Max 30k

最多 3 萬

Money Transferred

已轉帳

3-3. EXTERNAL SYSTEM

🔌 EXTERNAL SYSTEM 外部系統

- Pink Rectangle
- **External System or Legacy System** We **Integrate** with
- Usually **Encapsulate Details** We Don't Care
  - 通常會封裝（涵蓋）一些我們不需要在乎的細節。
  - 比如當我交給第三方金流時，我就不需要在乎他跟背後銀行的法規、串接流程。

Send Email

寄送 Email

SMTP Service Providers

Email Sent

Email 已寄送

Prey

祈禱

Legacy System

Miracle Happened

奇蹟已發生

[PRACTICE]
FLESH OUT (CONTD.)
5 MINS

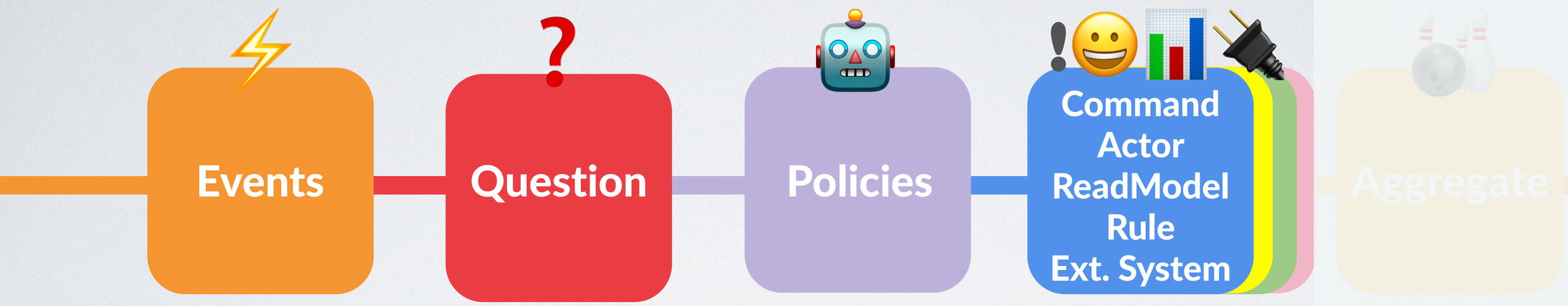
[PRACTICE] FLESH OUT (CONTD.) 5 MINS

- 思考每個 Command 與 Event 之間是否有些規則要遵守
- 找到 External System 的位置並貼上
- 注意！通常 External System 前面會接 Command 後面接 Event

EVENT STORMING - SOFTWARE DESIGN

Build Our System

Photo by Kevin Delvecchio on Unsplash



1. Explore Events

User Flow of Your Software

2. Raise Questions

Potential Threats of Your Design

3.1 React as Policies

The Automation Part of Your System

3.2 Flesh Out

Details to Design Your System

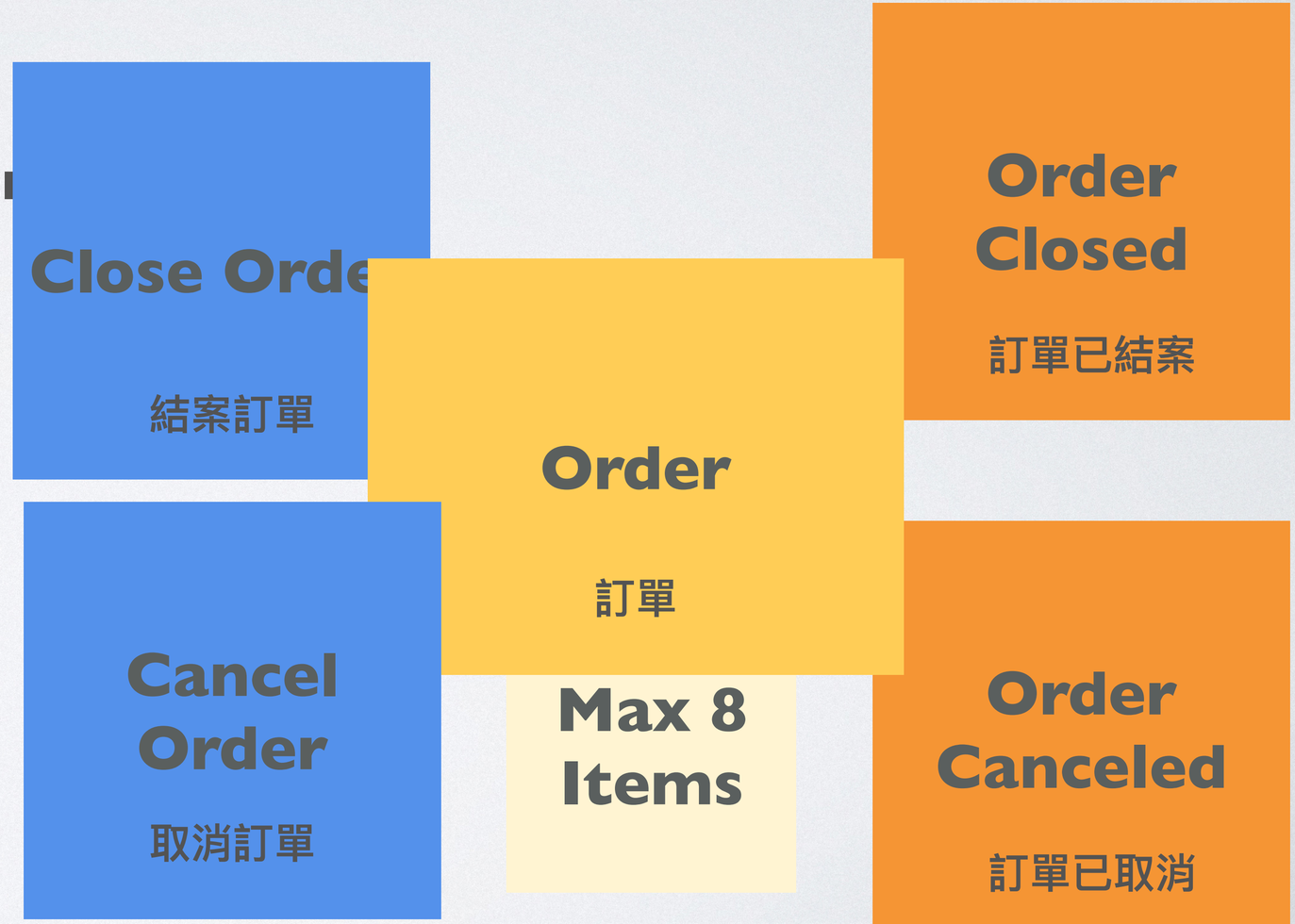
4. Design Aggregate

Main Characters (Models) of Your System

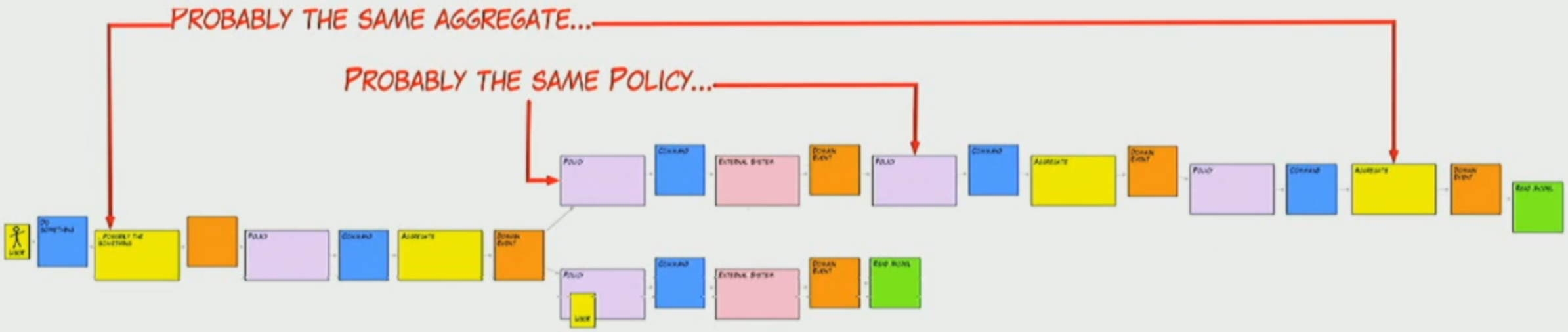
4-1. **FORM AGGREGATES**

🎳 AGGREGATE 負責人

- Coherent Business Model
- **Command-Aggregate-Event**
- Work as **State Machines**, or **Independen Components**
- Come with **Biz Rules**
- Break the Timeline
- Look for **Behavior** & **Responsibilities** Instead of Data
- **Postpone** Aggregate Naming



Symmetries on the timeline



Please Note: Timeline representation is suboptimal for OOD, but necessary to keep business stakeholders in the conversation.

Close Order
結案訂單

Order
訂單

Order Closed
訂單已結案

Cancel Order
取消訂單

Max 8 Items

Order Canceled
訂單已取消

Upgrade User
升級會員

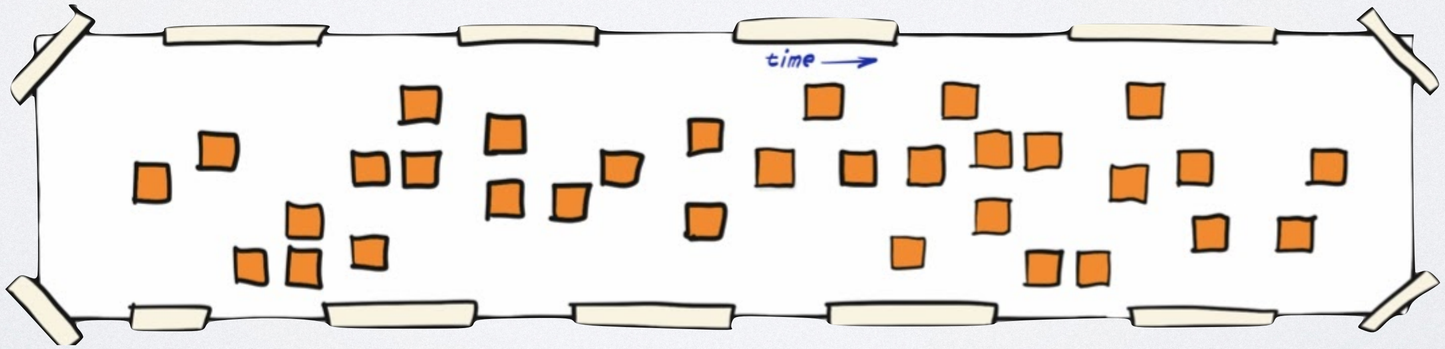
User
會員

User Upgrade
會員已升級

Blacklist User
黑名單會員

One Level at a Time

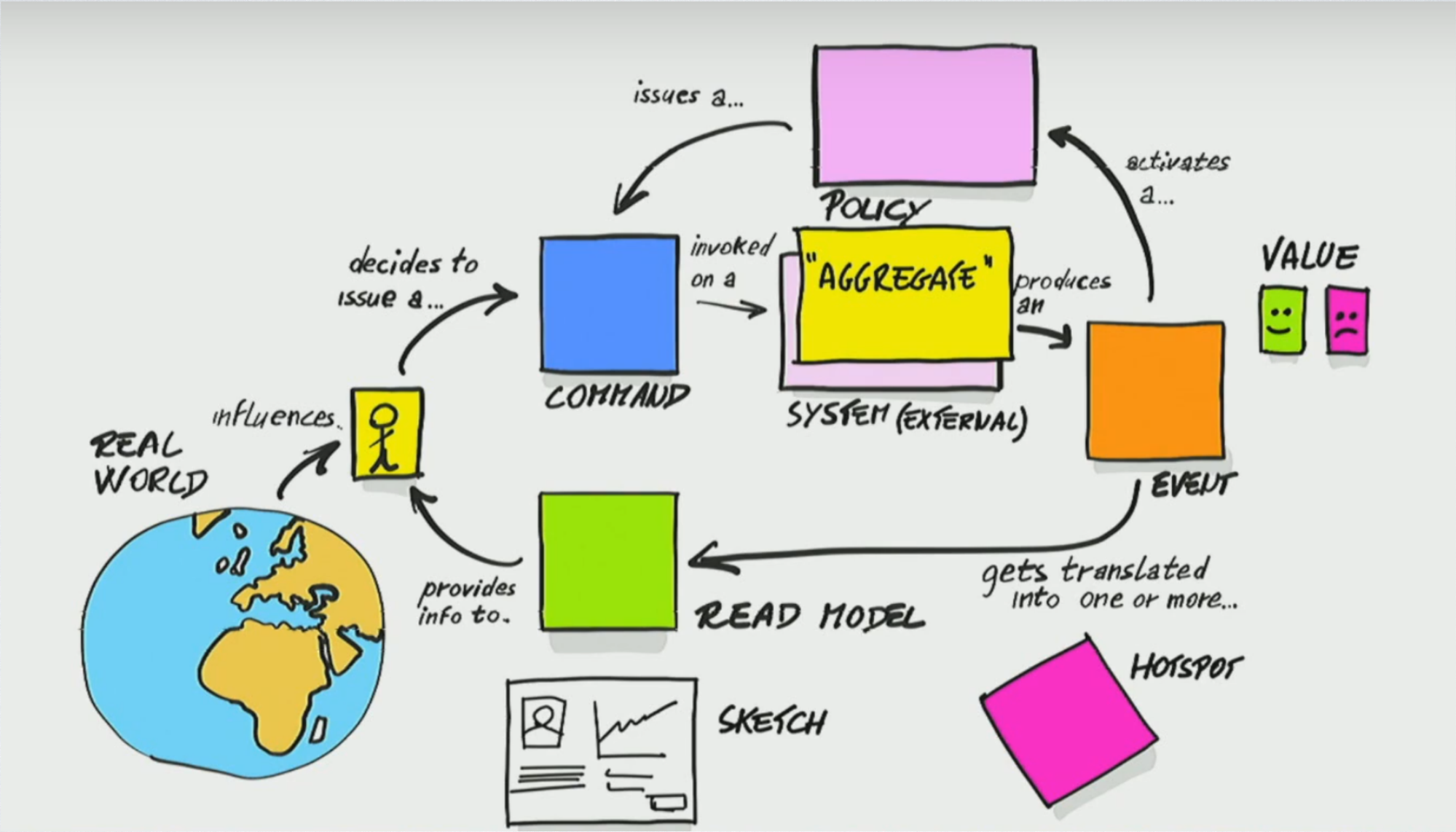
User Blacklisted
會員已被黑名單



[PRACTICE]
GROUP AGGREGATES
10 MINS

[PRACTICE] GROUP AGGREGATES 10 MINS

- 承接 Command，發出 Event。
- 可以看作是有狀態變化的「狀態機」或是一個獨立的元件。
- 專注在其「行為與職責」，一開始的命名可以廣泛一點，之後再求精確。不要執著於你的直覺。
- 可以破壞時間軸順序。


issues a...
POLICY
activates a...
VALUE
decides to issue a...
COMMAND
invoked on a
"AGGREGATE"
SYSTEM (EXTERNAL)
produces an
EVENT
influences.
REAL WORLD
provides info to.
READ MODEL
gets translated into one or more...
HOTSPOT
SKETCH

4-2. **BOUNDED CONTEXTS**

BEYOND UL:
CAN YOU TELL THE DIFFERENCE?

Account

帳戶

Account

帳號

Member

尊貴會員

BEYOND UL:
CAN YOU TELL THE DIFFERENCE?

Account Account VIP Member

帳戶 帳號 尊貴會員

No Difference

They Need Contexts!

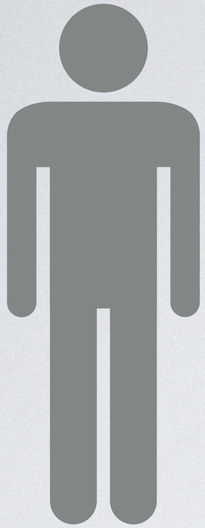
AFTER BC

Bank
Account 帳戶

FB
Account 帳號

伊莉
Member 尊貴會員

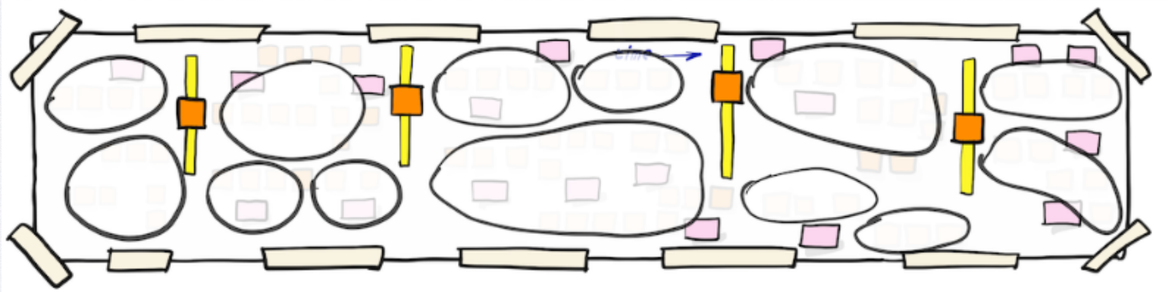
ANOTHER EXAMPLES: USER



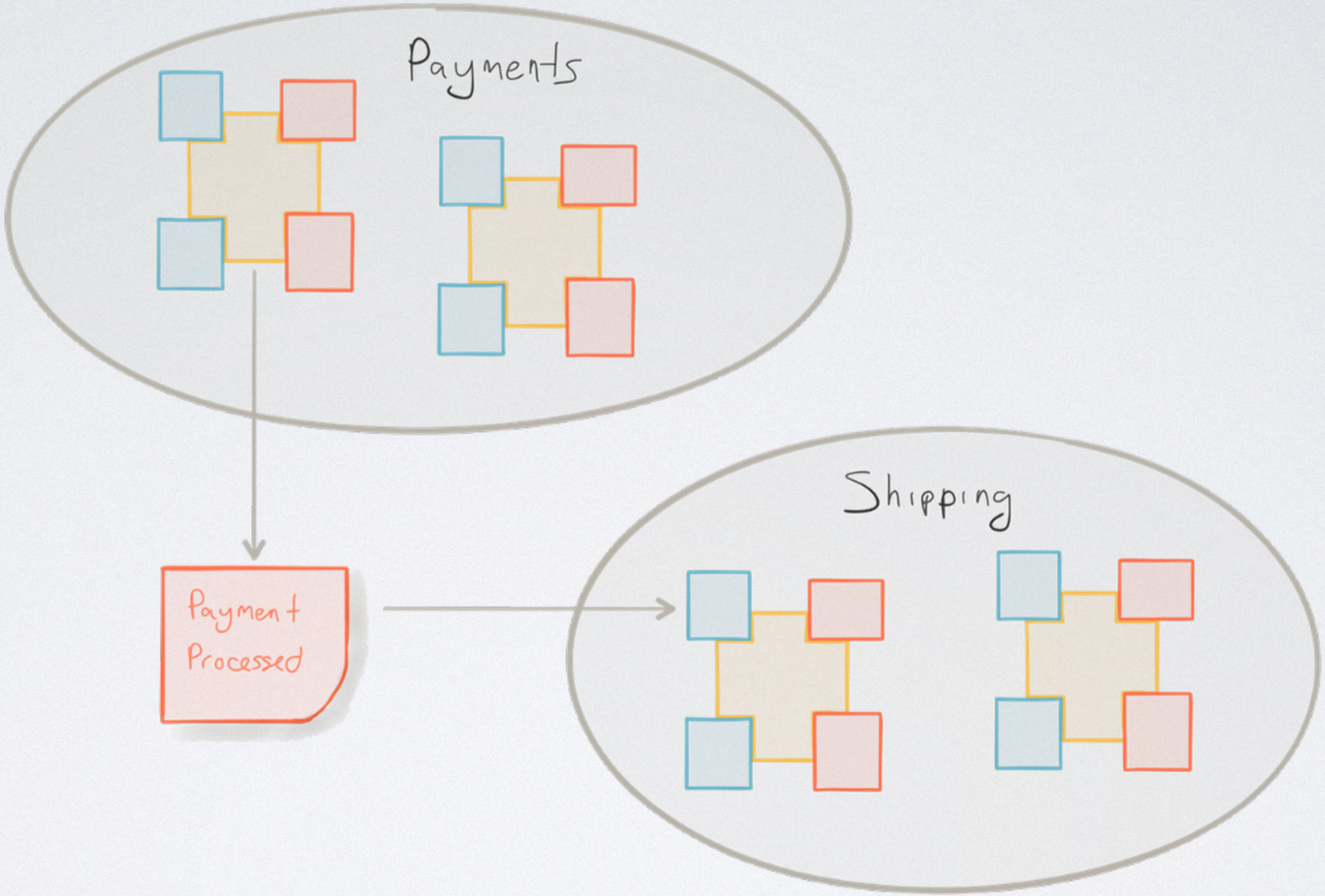
Payment	Member Service	Order
Payer	Member	Buyer

BOUNDED CONTEXT

- The Setting in Which a Word or a Statement Appears That Determines its Meaning
- The same concept might represent different thing in different contexts!
- A Candidate for Your Module or System
- Natural Business Boundary



Emerging bounded contexts after a Big Picture EventStorming.


Payments
Payment Processed
Shipping

[PRACTICE]
BOUNDED CONTEXTS
5 MINS

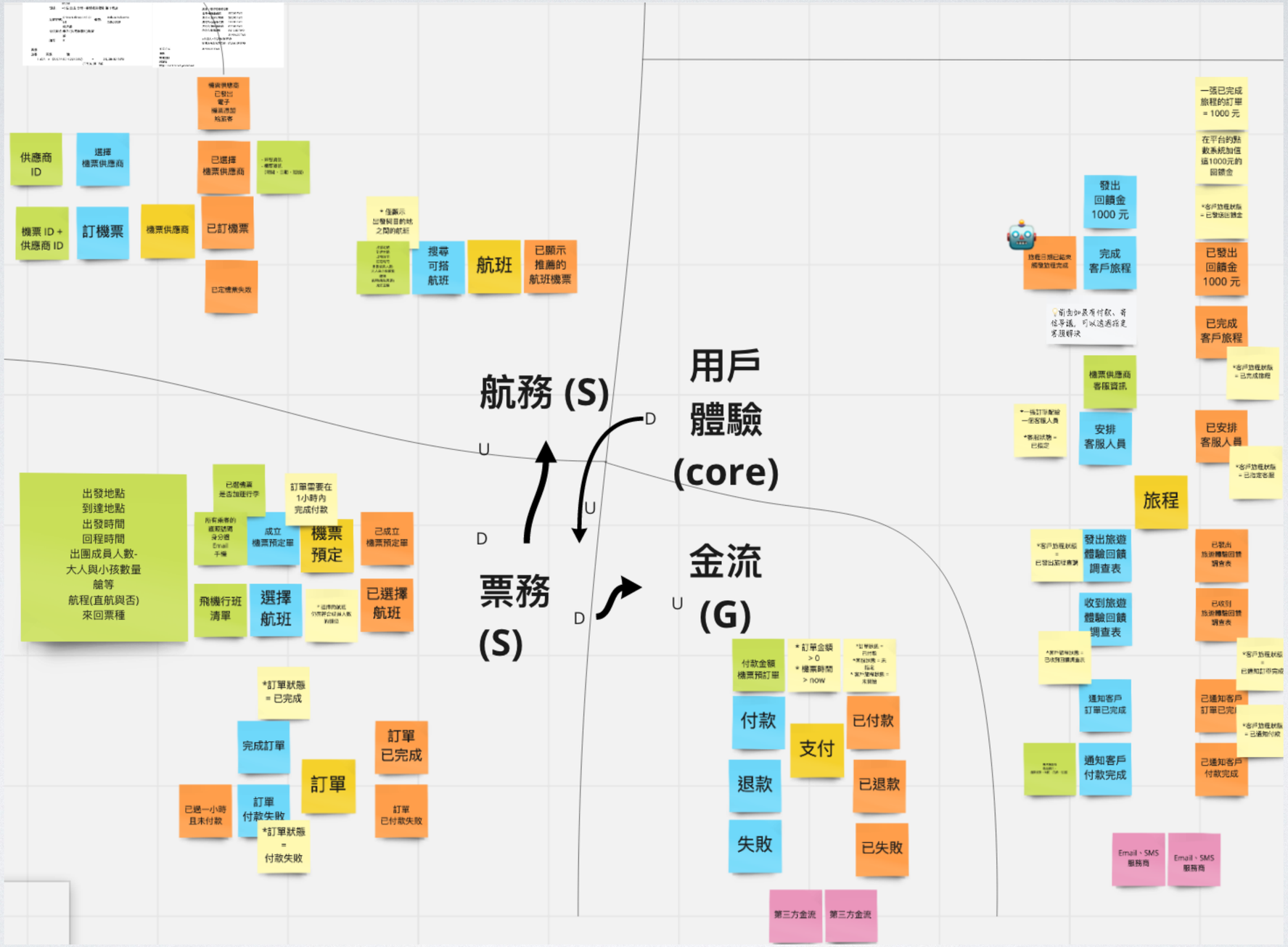
[PRACTICE] BOUNDED CONTEXTS

5 MINS

- 如果你覺得這幾張 Aggregate 解決的問題很相似，就把他們 Group 起來。 EX: 訂單與折扣活動可以分類到「購買」Context、會員與會員群組可以分到「會員管理」Context
- 根據內聚力與業務相依性初步勾勒邊界
- 不需要太過糾結未知的範疇，先以目前捕捉的為主
- 有時間的話，可以用箭頭表示兩者間的依賴關係

CAVEAT

- 不用分到太細
- 通常可以設計一個大系統，在未來再拆分出來


航務 (S)
用戶
體驗
(core)
票務
(S)
金流
(G)
U
D
U
D
D
U
供應商
ID
選擇
機票供應商
已選擇
機票供應商
機票 ID +
供應商 ID
訂機票
機票供應商
已訂機票
已定機票失敗
搜尋
可搭
航班
航班
已顯示
推薦的
航班機票
出發地點
到達地點
出發時間
回程時間
出團成員人數-
大人與小孩數量
艙等
航程(直航與否)
來回票種
成立
機票預定單
機票
預定
已成立
機票預定單
訂單需要在
1小時內
完成付款
飛機行班
清單
選擇
航班
已選擇
航班
*訂單狀態
= 已完成
完成訂單
訂單
訂單
已完成
已過一小時
且未付款
訂單
付款失敗
*訂單狀態
=
付款失敗
訂單
已付款失敗
付款金額
機票預訂單
* 訂單金額
> 0
* 機票時間
> now
付款
支付
已付款
退款
已退款
失敗
已失敗
第三方金流
第三方金流
一張已完成
旅程的訂單
= 1000 元
在平台的點
數系統加值
值1000元的
回饋金
發出
回饋金
1000 元
完成
客戶旅程
已發出
回饋金
1000 元
已完成
客戶旅程
機票供應商
客服資訊
安排
客服人員
已安排
客服人員
旅程
發出旅遊
體驗回饋
調查表
收到旅遊
體驗回饋
調查表
通知客戶
訂單已完成
通知客戶
付款完成
已通知客戶
付款完成
Email、SMS
服務商
Email、SMS
服務商

RECAP

- No One Knows Everything
- Communication & Collaboration is the Key
- Understand your **Domain Knowledge**
- Events are **Evidence**
- Don't Miss the Automation Part (Process)
- Aggregates are Main Characters of Your System
- Bounded Contexts Make **Business Boundary**

FURTHER STUDIES

- Embed event storming in your agile toolset

"To communicate effectively, the code must be based on the same language used to write the requirements—the same language that the developers speak with each other and with domain experts."

要有效溝通，程式碼必須基於跟規格書上同樣的語言 — 那個可以讓開發人員與領域專家彼此溝通的語言。

–Eric Evans

THANK YOU

REFERENCE

- https://unsplash.com/
- Alberto - Introducing Eventstorming
- Alberto Brandolini - 50,000 Orange Stickies Later - YouTube
- Modelling Reactive Systems with Event Storming and Domain-Driven Design
- Think in Domain-Driven Design
- A facilitators recipe for Event Storming
- DDD实践手册：事件风暴实践
- https://github.com/ddd-crew/eventstorming-glossary-cheat-sheet
- https://developer.ibm.com/languages/java/tutorials/reactive-in-practice-1/

CONTACT ME

- Email: fong1143@gmail.com
- Telegram: @Fong_L
- Join FB Club: Domain Driven Design(DDD Taiwan)