Przyjęte założenia w Event Storming

Co zyskujemy dzięki Event Storming?

- Precyzyjne modele procesów biznesowych
- Jednolity język do rozmowy I w kodzie
- Wyraźnie oznaczone HotSpot-y
- Granice Bounded Contextów
- Model do pisania kodu
- Wsad do utworzenia Epics, Features, User Stories, Tasks
- Event mapuje się na Product Backlog Item / User Story

Jakie założenia robimy w naszym Event Storming?

- **1 etap to Event Storming Big Picture**
- **2 etap to Process Level + Design Level** robiony w jednej sesji
- **Utworzone modele podlegają ciągłemu refactoringowi**
- Realizujemy warsztaty w miro, bo nie ma innego wyjścia
- Event mapuje się na Product Backlog Item / User Story
    - Kryteria Akceptacji pisane zgodnie z notacja BDD (Given, When, Then) moza zakodować na modłe testów automatycznych

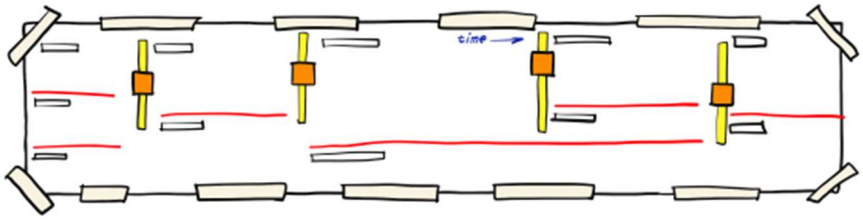
Kto uczestniczy w warsztacie Event Storming Big Picture?

- **Ekspert Domenowy** - główne źródło zdarzeń domenowych
- **Osoba decyzyjna** – pomaga rozwikłać HotSpot-y, doprecyzować
- **Modelarz** – rysuje na miro
- **Facylitator** – trzyma reguły event Stormingu

Jak realizujemy Big Picture?

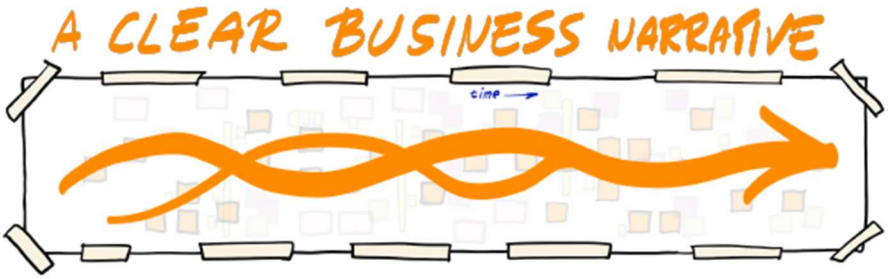
- Tworzymy **Legendę** na Boardzie
- Kleimy **Zdarzenia Domenowe**
    - Czasownik czas przeszły (“Wystawiono Fakturę”)
    - Reprezentują istotną zmianę stany procesu
- **Chaotyczna Eksploracja**
- Porządkujemy je według **Linii Czasu** i usuwamy **duplikaty**

- Dodajemy strukturę
    - Znajdowanie **kluczowych zdarzeń** I wokół nich rozbudowa procesów -
    - **Grupowanie** zdarzeń w procesy / pod procesy
    - Dodanie **SwimLine-ów**



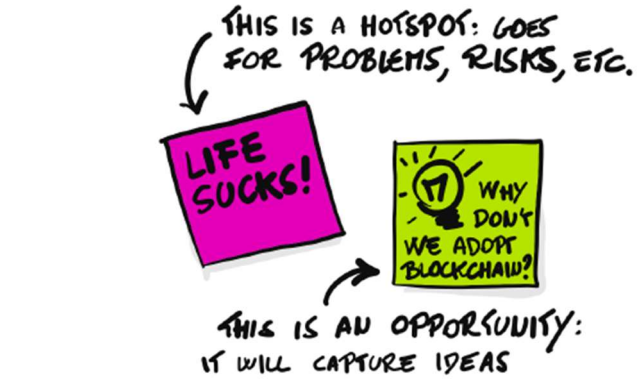
You can combine different approaches in something more sophisticated, but it's hard to define the structure upfront.

- **Debugujemy i Poprawiamy**:
    - Czytamy procesy
    - Co musiało się zadziać, żeby zaszło zdarzenie X, Czy po zdarzeniu X zajdzie zdarzenie Y, szukamy rozgałęzień zdarzeń,
- Opowieść od końca
- Dodajemy **HotSpot-y**
- Dodajemy **Aktorów i Zewnętrzne Systemy**
- W tym momencie chcemy mieć **Clear Business Narrative:**

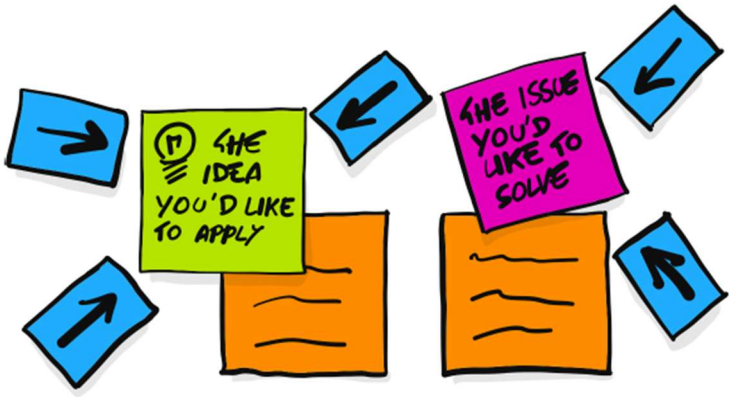


A clear business narrative, delivering value to the key players in the system. This is what I am expecting to "see" at the end.

- Dodajemy **Problemy** i **Okazje** 15 minut, używamy **arrow voting** do wyboru kluczowych miejsc



Balancing problems and opportunities: a purple problem flood might seem overwhelming, but ...we have a lot of green ideas to the rescue!



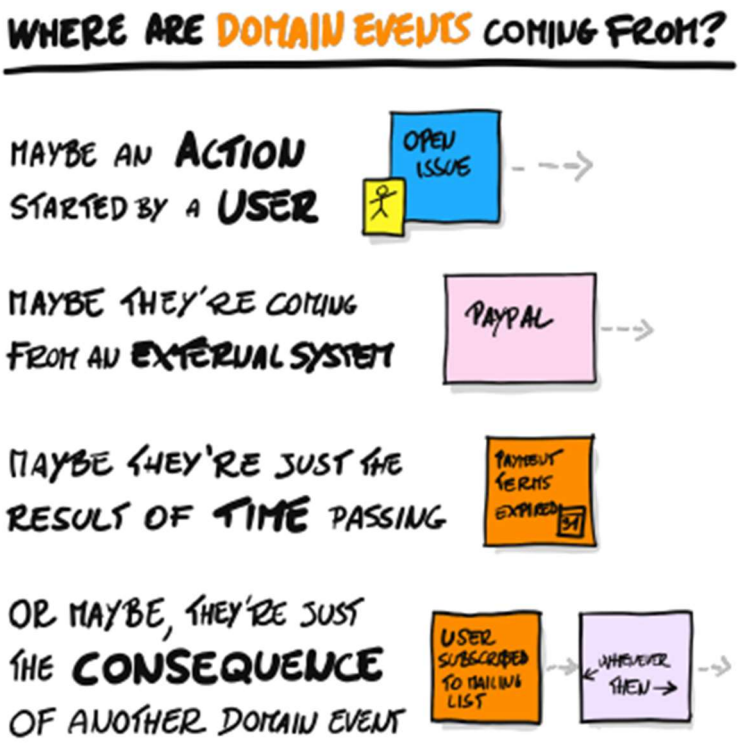
Arrow voting in action, pick your target!

Kto uczestniczy w warsztacie Process Level + Design Level?

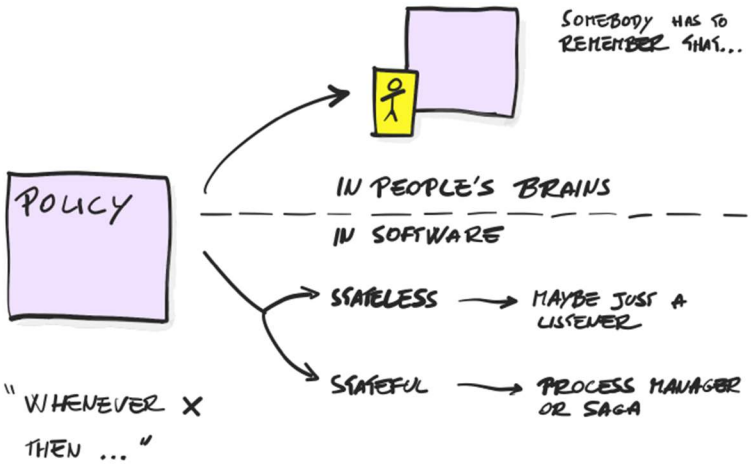
- Modelarz
- Facylitator
- Developer
- Architekt
- Ekspert domenowy - Reguły

Jak realizujemy Process Level + Design Level?

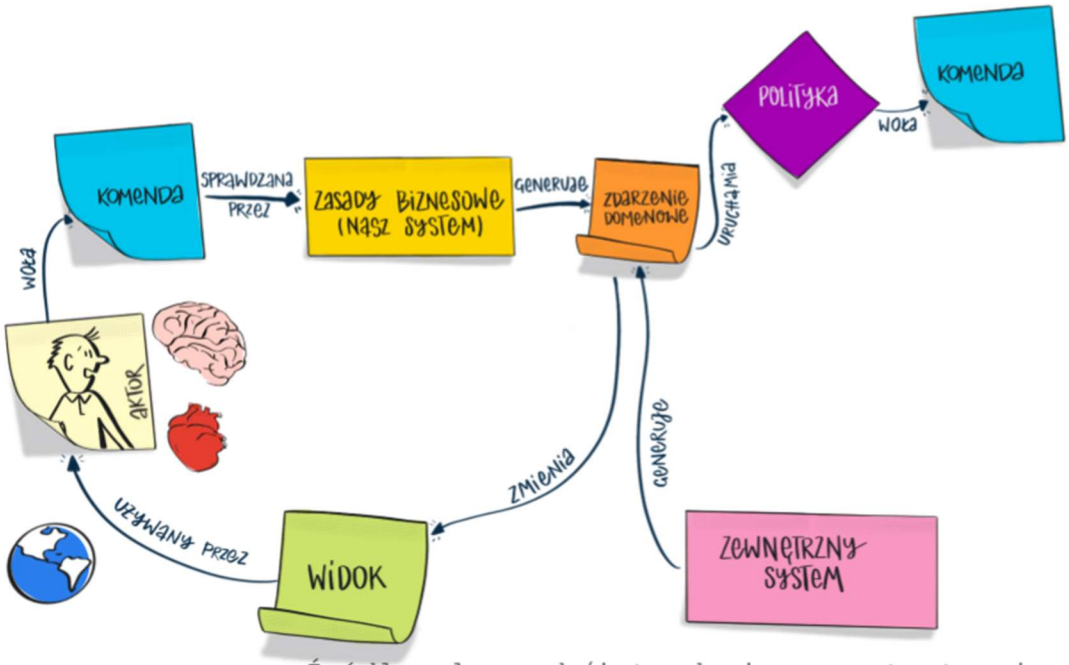
- Update-ujemy **Legendę** na Boardzie
- Nazywamy **Procesy Biznesowe**
- Uwzględniamy **Widoki**
- Dodajemy **Komendy**



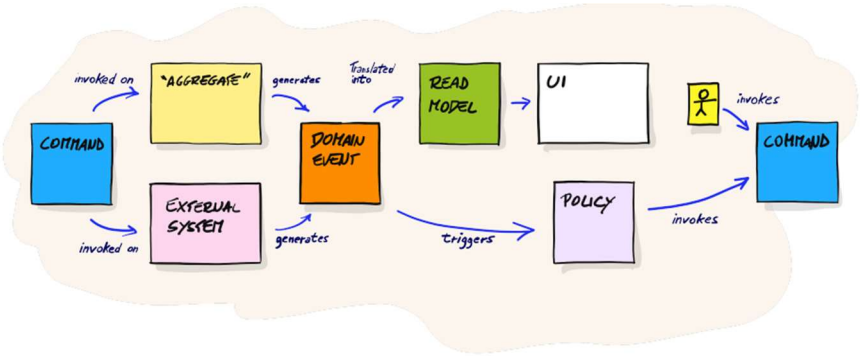
- Refaktorujemy Modele Procesów Biznesowych
    - Np dodajemy / usuwamy zdarzenia
    - Debugujemy I Poprawiamy Flow
- Dodajemy **Reguły**
- Wyznaczamy granice **Bounded Context-ów**
- Rozwiązujemy **Hot Spoty**
- Dodajemy **Polityki**
    - To reaktywna logika na zdarzenie
    - Brakujący element między zdarzeniem a komendą

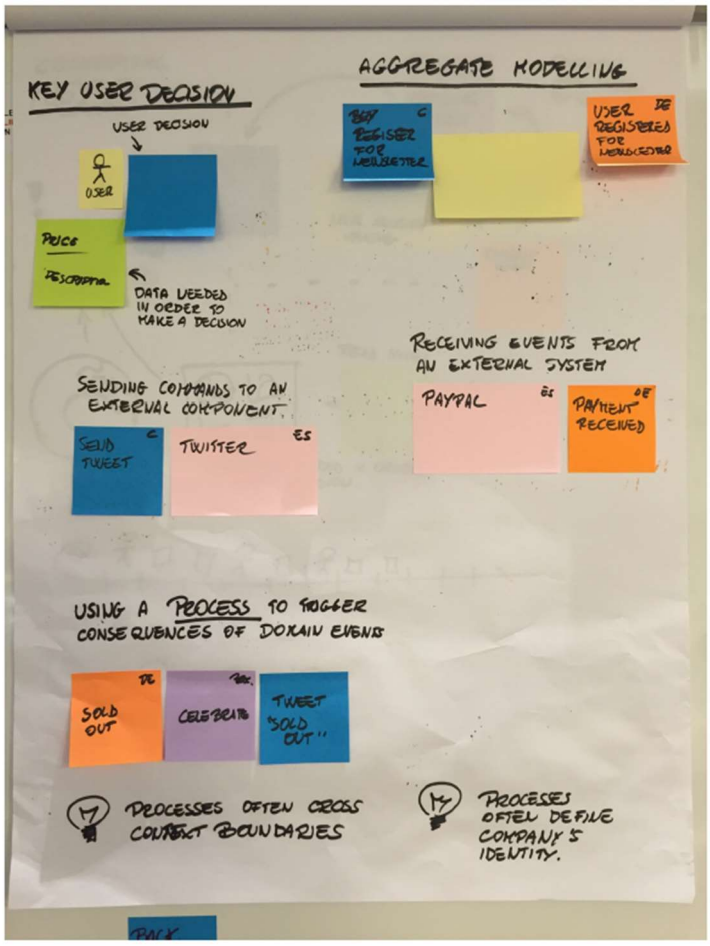


Jak wyglądają elementy ES Design Level razem?



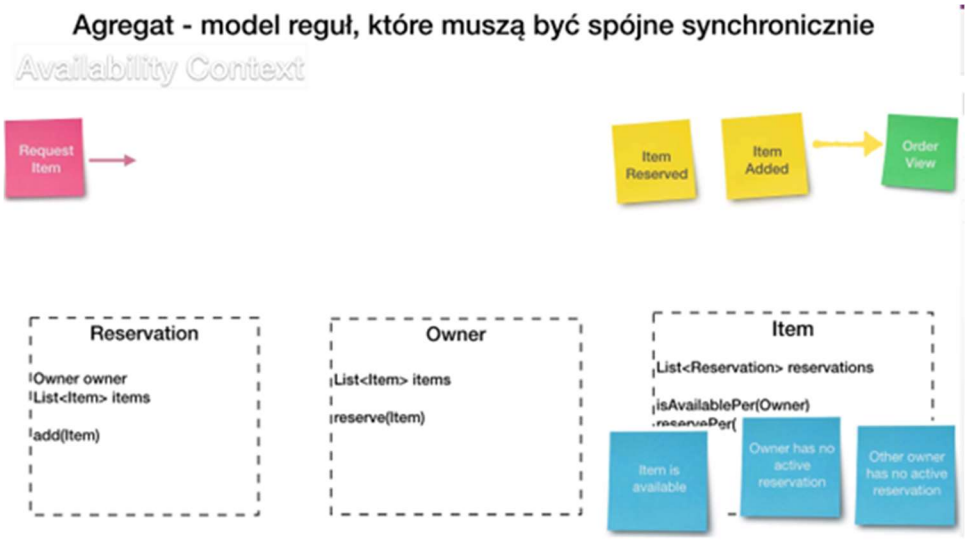
Źródło: leanpub/introducing_event_storming


invoked on
"AGGREGATE"
generates
translated into
READ MODEL
UI
invokes
COMMAND
DOMAIN EVENT
EXTERNAL SYSTEM
POLICY
invokes
COMMAND
invoked on
generates
triggers


KEY USER DECISION
USER DECISION
USER
POLICY
DESCRIPTION
DATA NEEDED IN ORDER TO MAKE A DECISION
AGGREGATE MODELLING
BUY
REGISTER FOR NEWSLETTER
USER REGISTERED FOR NEWSLETTER
RECEIVING EVENTS FROM AN EXTERNAL SYSTEM
PAYPAL
PAYMENT RECEIVED
SENDING COMMANDS TO AN EXTERNAL COMPONENT
SEND TWEET
TWITTER
USING A PROCESS TO TRIGGER CONSEQUENCES OF DOMAIN EVENTS
SOLD OUT
CELEBRATE
TWEET "SOLD OUT"
PROCESSES OFTEN CROSS CONTEXT BOUNDARIES
PROCESSES OFTEN DEFINE COMPANY'S IDENTITY.

Jak definiujemy Agregaty w Event Storming Design Level?

- Określamy **Agregaty** – klasy, które trzymają te dane i **reguły**, które musza być spójne atomowo. Inne forma modelu, bez timeline-a



Z jakimi decyzjami architektonicznymi może nam pomóc ES Design Level?

- Kod jest pisany zgodnie z wymaganiami
- Dostarczy modeli do budowy systemu
- Używamy w kodzie **Ubiquitous Language** (Wszechobecny język)
- CQRS (**Widoki** I **Komendy** na modelach)
- Podział aplikacji na Moduły i wybór architektury per moduł w Modularnym Monolicie (granice **Bounded Context**-ów)
- Gdzie Rich Domain Model (dużo **Reguł + Agregat**), Gdzie Anemic Model - CRUD (**mało Reguł**),
- Jak forma komunikacji synchroniczna (REST API), asynchroniczna (Architektura sterowna zdarzeniami + Broker), dwustronna (WebsSokety) widać z **Modeli Procesów**, **Integracji**, **Polityk, Komend**
- Gdzie zaimplementować obsługę długo żyjącego asynchronicznego handlera wiadomości - czyli Sagę (widać z modelu **Polityki**, np. Polityka obsługująca przyznanie premii pracownikowi)

Jak to wygląda to razem?