DOMAIN DRIVEN DESIGN EUROPE 2023

DDD in large product portfolios

Andreas Pinhammer

Key takeaway n°1
Starting from a greenfield in a complete environment can be a simple and effective way of experimenting with DDD

Key takeaway n°2
Platforms that are not specific to a given product limit the mental workload of teams and minimize duplication

Exposition

Une grande compagnie d'assurance allemande
2.000+ assets techniques

Rising action

Bad Time To Market

Doesn't attract young people

Climax

Height teams

Cross-functionnal

Experiment Agile

Experiment DDD

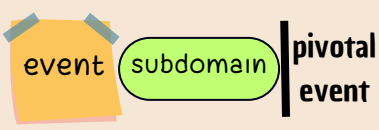
Experiment DevSecOps

Build a MVP

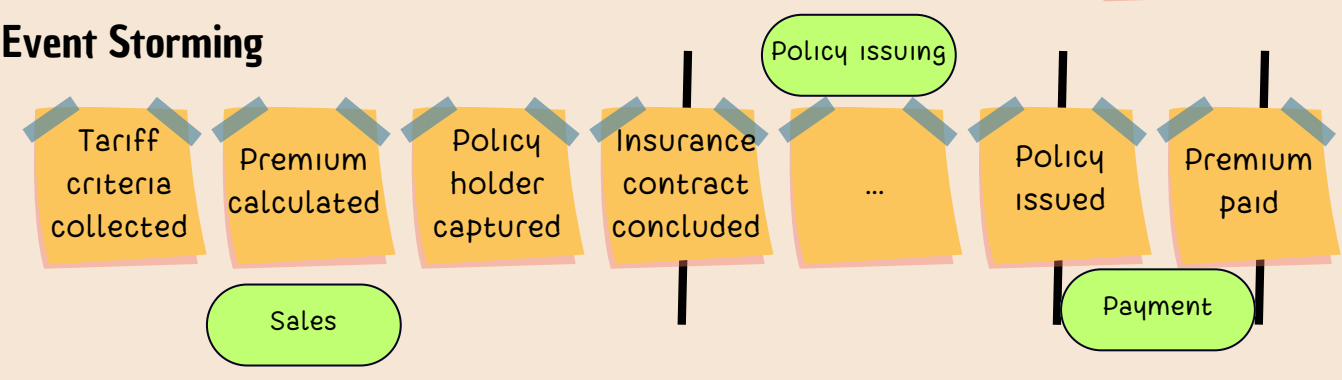
Product civil liability

From scratch

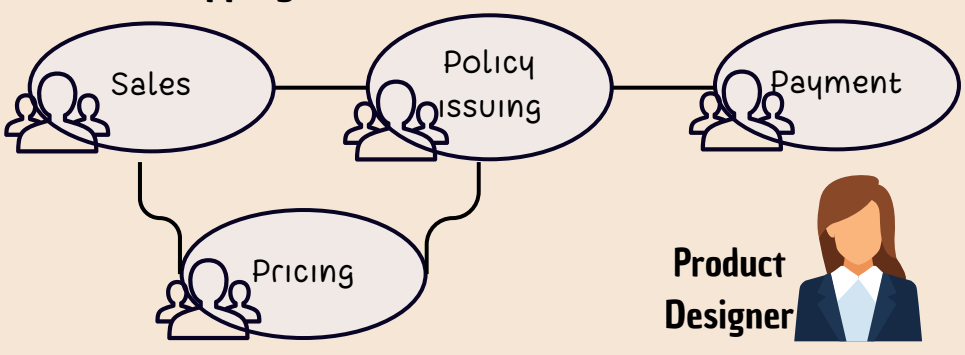
Using the Inverse Conway Maneuver
Align the organisation with the desired domain



1. Event Storming



2. Context Mapping



1 subdomain = 1 team

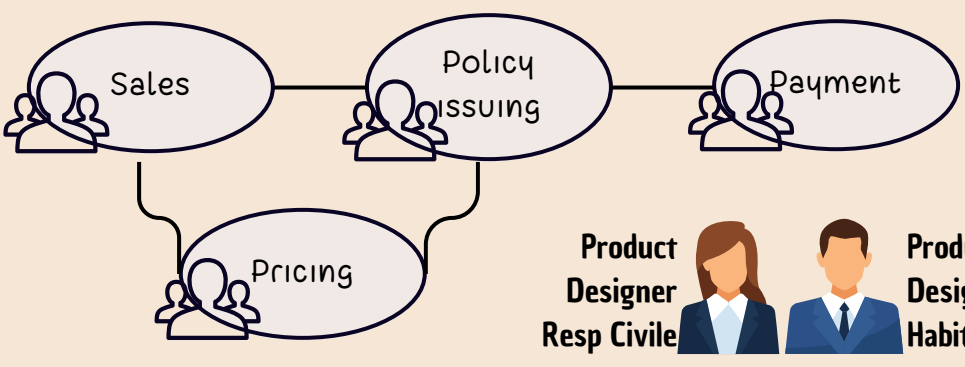
Product Designer

The product designer promotes a unified vision across all teams

1 year later, the MVP is up and running
Choice to go on a second product (furniture insurance), to test the scalability of the model

Repeat steps 1 and 2

Similar Event Storming --> Adding a dedicated product designer



Product Designer Resp Civile

Product Designer Habitation

3 problems

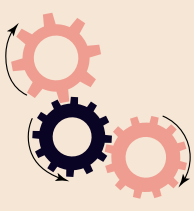
Backlog alignment
multi-product backlog difficult to manage

Product feedback
Designers far from feedback
Low team impact

Mental load
Each new product adds to the teams

3. Retrospective

Why do new products add to developers' mental load ?



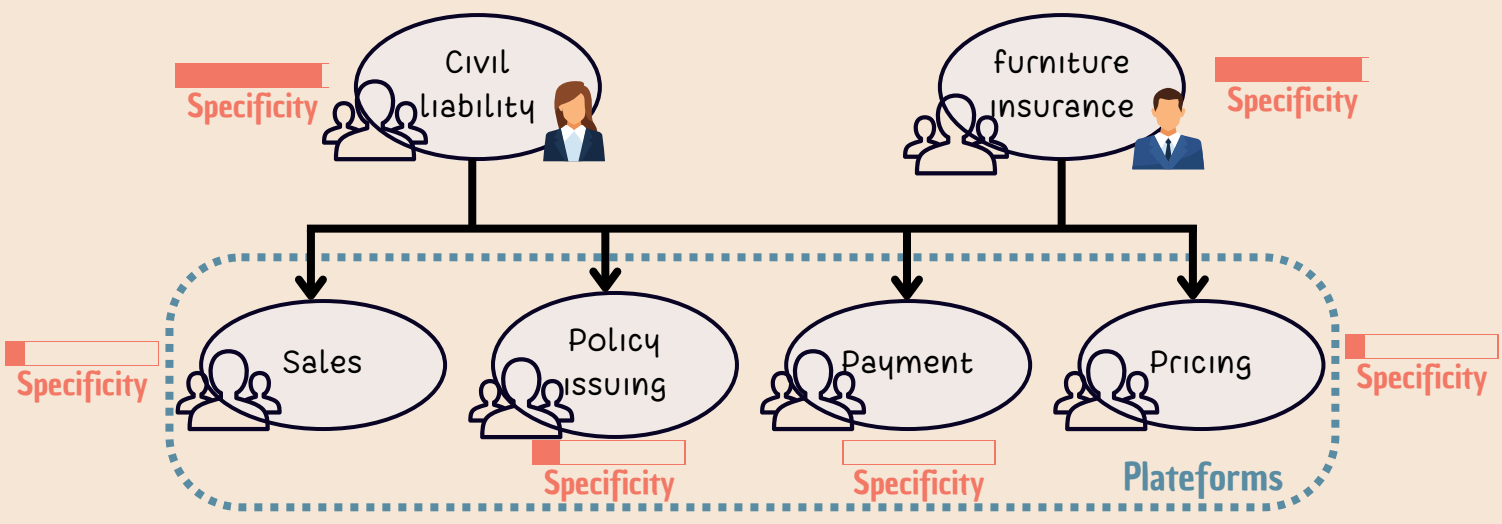
Similar processes, but...

Different vocabularies

Different customer needs

Use generic platforms — A product is a subdomain

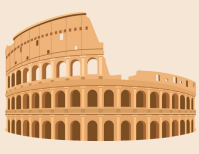
Falling action



Bonus story

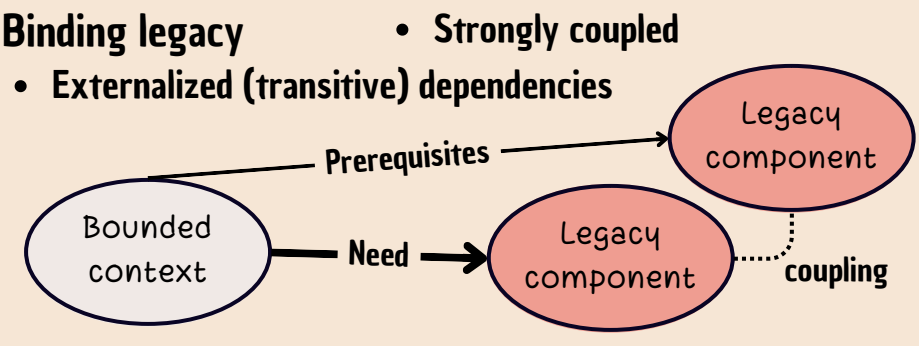
How can these new products ...

... be integrated with the old legacy?



Binding legacy

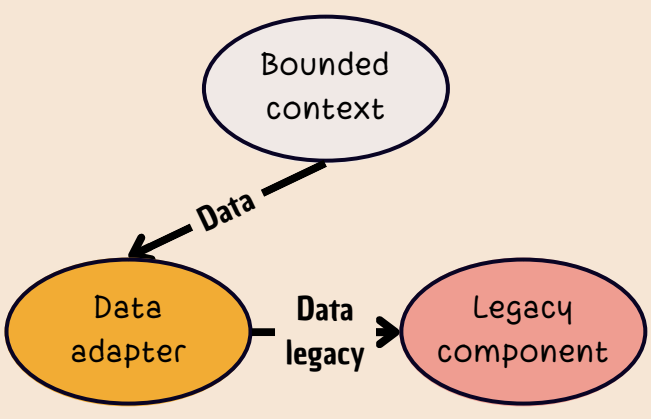
- Strongly coupled
- Externalized (transitive) dependencies



Setting up an "Ice Wall"

Assembly of Anti Corruption Layers (ACL) between legacy and DDD components. 2 types of adapters :

Data Supply ACL



Pseudo Bounded Context ACL