



*Devoxx* France 2023

---

Rendons le  
DDD aux devs !

Arnaud THIEFAINE  
Dorra BARTAGUIZ

AROLLA

*Sondage à main levée*

**Qui connaît DDD ?**

**Qui a essayé de  
l'appliquer ?**

**Qui arrive  
réellement à  
l'appliquer ?**



# Pourquoi apprendre DDD ?

Parce que c'est trop stylé !

Parce que je veux faire comme celles et ceux qui en parlent

Pour faire bien sur mon CV

Parce que c'est peut-être utile en fin de compte ?

# Ce qu'on reproche au DDD

Je ne suis pas architecte donc je n'ai pas eu l'opportunité de creuser

Je n'ai pas accès aux discussions stratégiques de l'organisation

L'organisation et les équipes sont trop hermétiques pour en faire

Ca ne redescend pas jusqu'au code

Ça sert surtout à faire des diagrammes bullshit



**DDD**  
**Buzz Word ?**

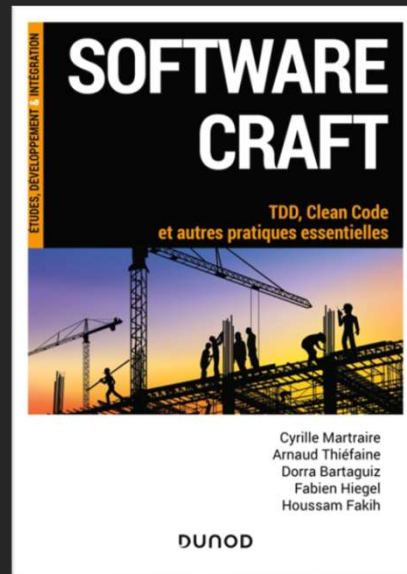


Qui?

# Arnaud THIEFAINE

Coach / Formateur  
Co-auteur « Software Craft »

Twitter ArnaudThiefaine  
Github athiefaine



# Dorra BARTAGUIZ

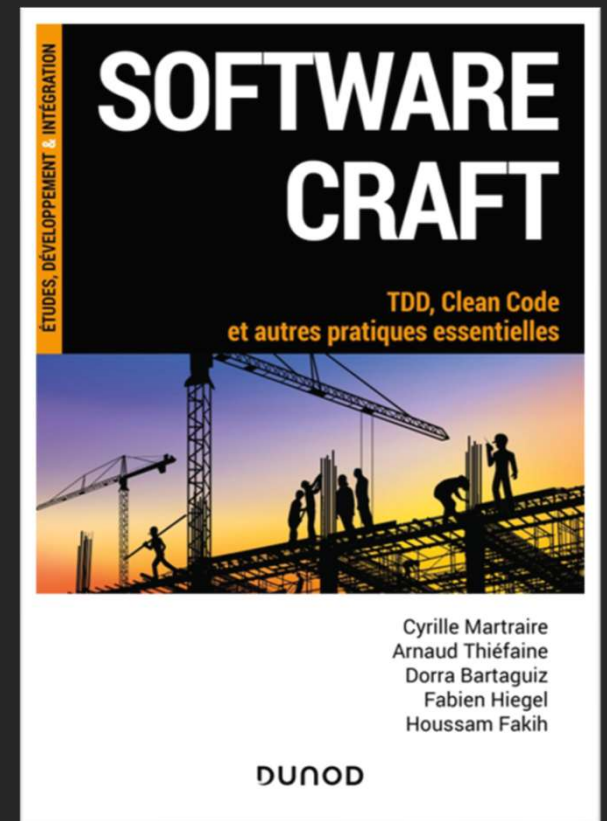
VP Tech @Arolla  
Co-auteure & illustratrice « Software Craft »

Twitter DorraBartaguiz  
Github iAmDorra



<https://www.arolla.fr/training/>

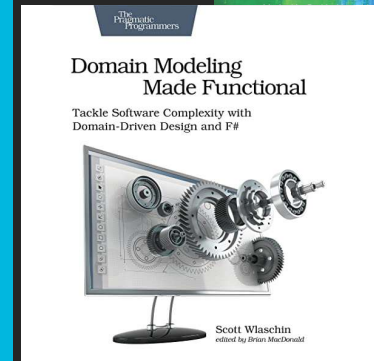
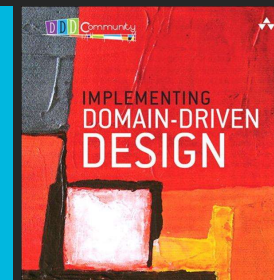
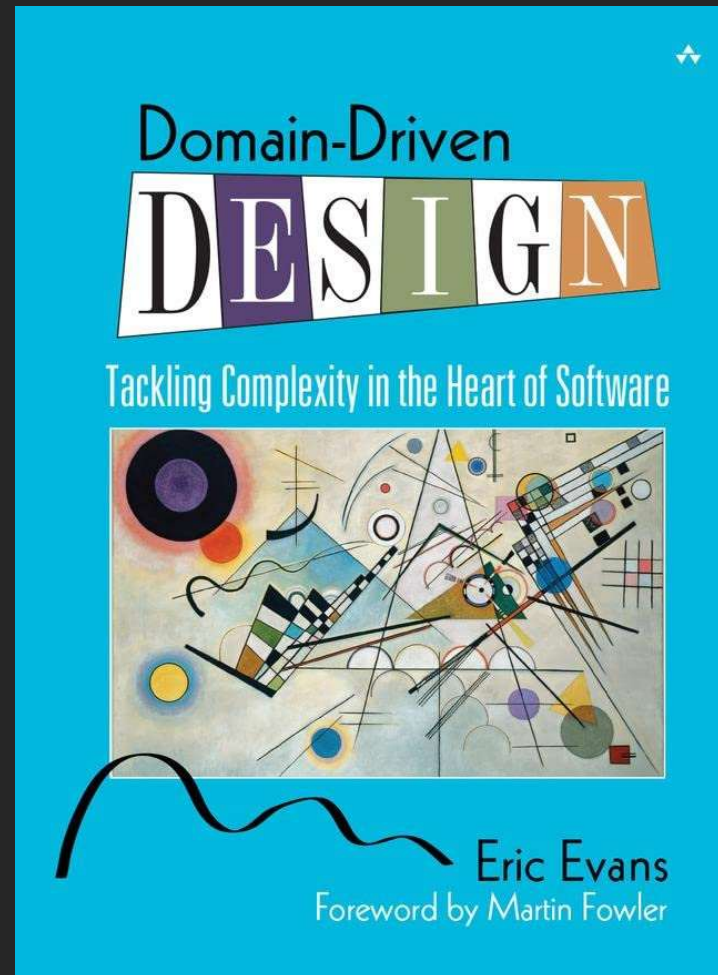
**Essai à blanc**  
**<https://kahoot.it>**  
**3592823**



DDD



20 ans

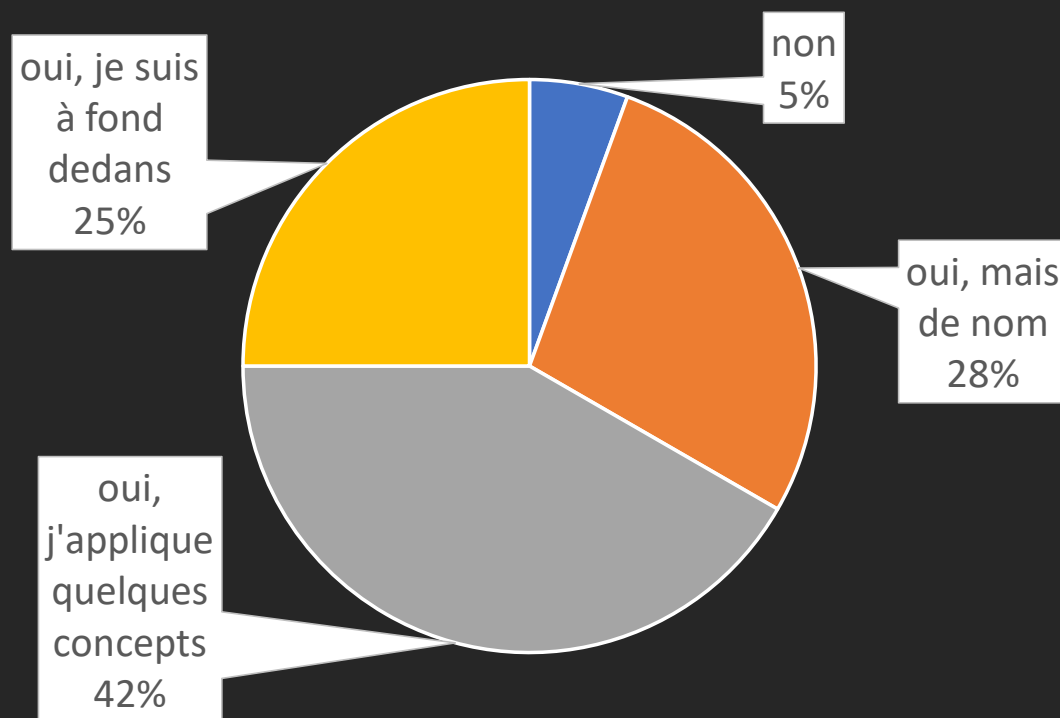




# Enquête



## Connaissez-vous DDD?





~56%

Apprécient DDD

~63%

Trouvent difficile à  
appliquer



## Votre rapport à DDD - Domain Driven Design

Durée estimée d'exécution : 10 min

1. Est-ce que vous connaissez DDD - Domain Driven Design ?

- ☐ non
- ☐ oui, mais de nom
- ☐ oui, j'applique quelques concepts
- ☐ oui, je suis à fond dedans

2. A quel point vous appréciez DDD ?

☆ ☆ ☆ ☆

3. A quel point vous trouvez que c'est difficile à appliquer ?

☆ ☆ ☆ ☆



# Ce qu'on reproche au DDD

Je suis pas architecte donc je suis pas eu l'opportunité de creuser

Je n'ai pas accès aux discussions stratégiques de l'organisation

L'organisation et les équipes sont trop hermétiques pour en faire

Ca redescend pas jusqu'au code

Ca sert surtout à faire des diagrammes bullshit

Rendons *DDD* aux devs !

## SUMMARY



### 1<sup>e</sup> partie

Kata de modélisation (Theater Kata)

Value Object au secours des primitive-obsessions

Séparation du métier de la technique

pattern sandwich

Identification des Bounded Contexts

### 2<sup>e</sup> partie

Séparation du métier et de la technique

archi hexa

Protéger les invariants (Aggregate, Entity, Value Object)

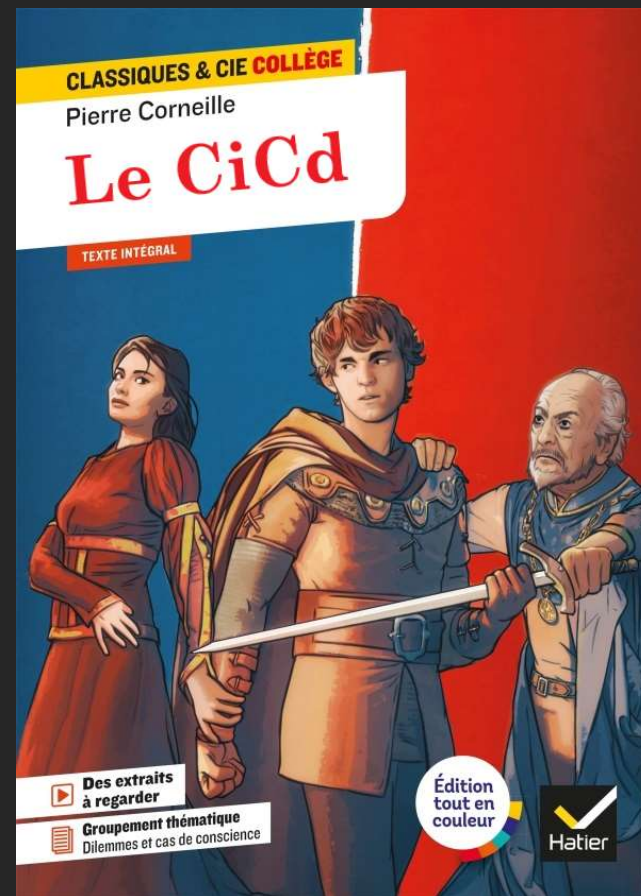
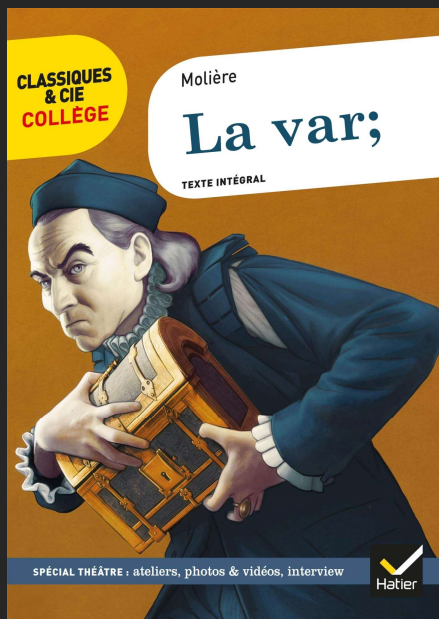
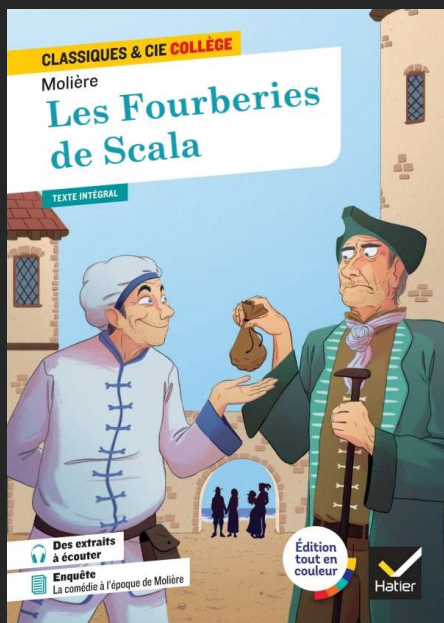
Service & repository

Découpage en Bounded Contexts





## Barnabé au Théâtre de Paris





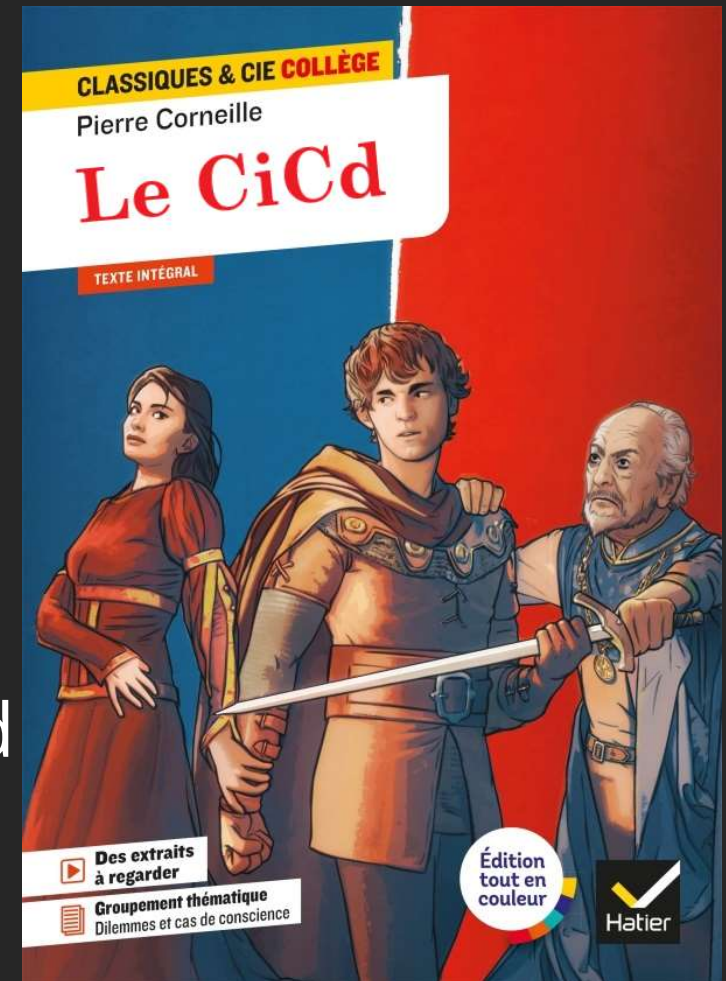
## Barnabé « The CiCd »



50% VIP



35€ catégorie standard  
+50% en premium





Éric réserve pour quatre



Catégorie premium

Sièges côte-à-côte

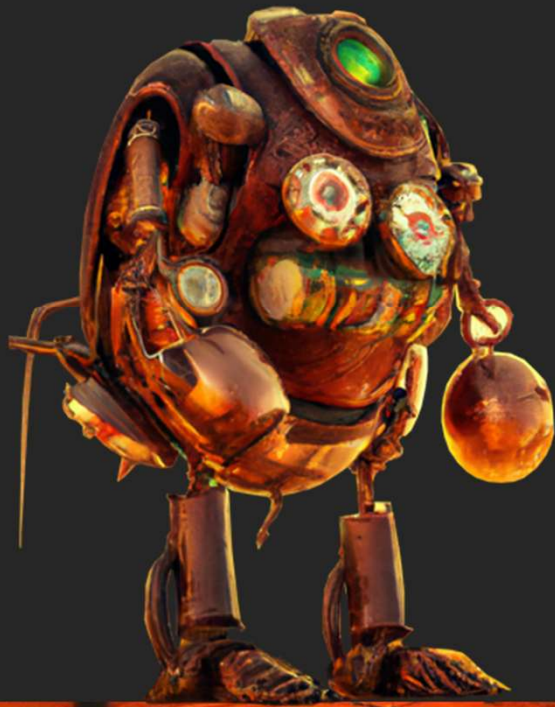


Carte abonnement



Tickets





**Passons au code !**



# Extract Value Type

1. Extract method from primitive value
2. Move to a new target class
3. Convert from static to non-static
4. Extract primitive value as field initialized in constructor
5. In constructor, extract parameter for the value

## Extract Value Type operation

1. Wrap the result of operation in value type
2. Extract method from value type expression
3. Wrap each parameter as value type
4. Extract wrapped value types as parameters
5. Convert from static to non-static (this action moves the method in the value type)

# Pricing arithmetics



**DSL**

# Pricing arithmetics



$\times$

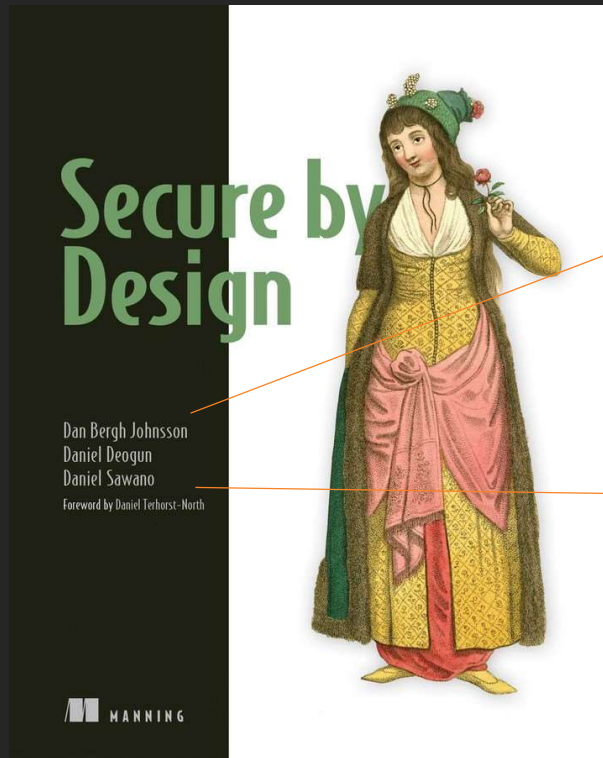


$+$





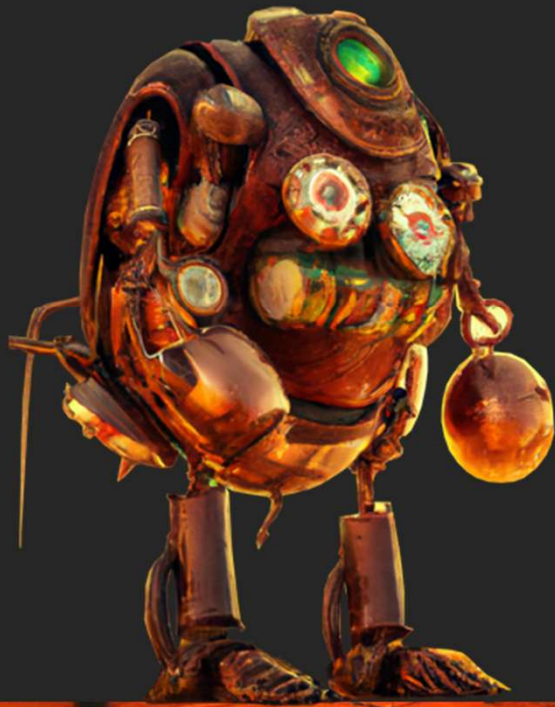
# Secure by design



Dan  
Daniel  
Daniel

La Théorie  
du complot  
pour  
les nuls





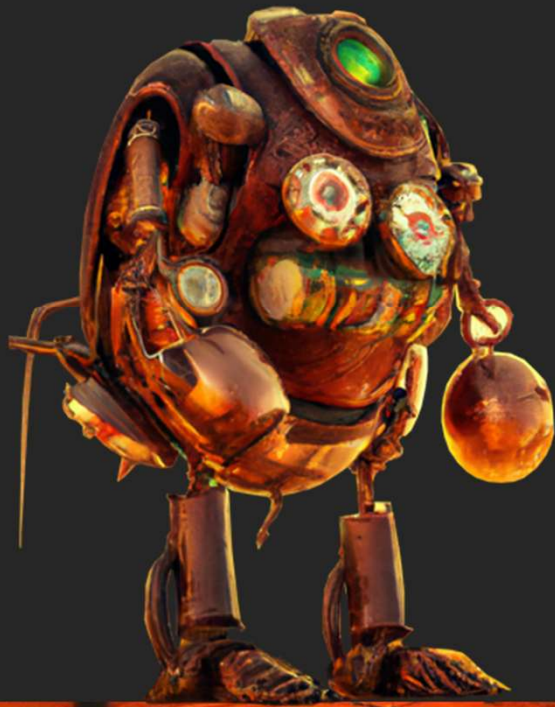
**Revenons  
au code !**

Refactoring

# Pattern Sandwich








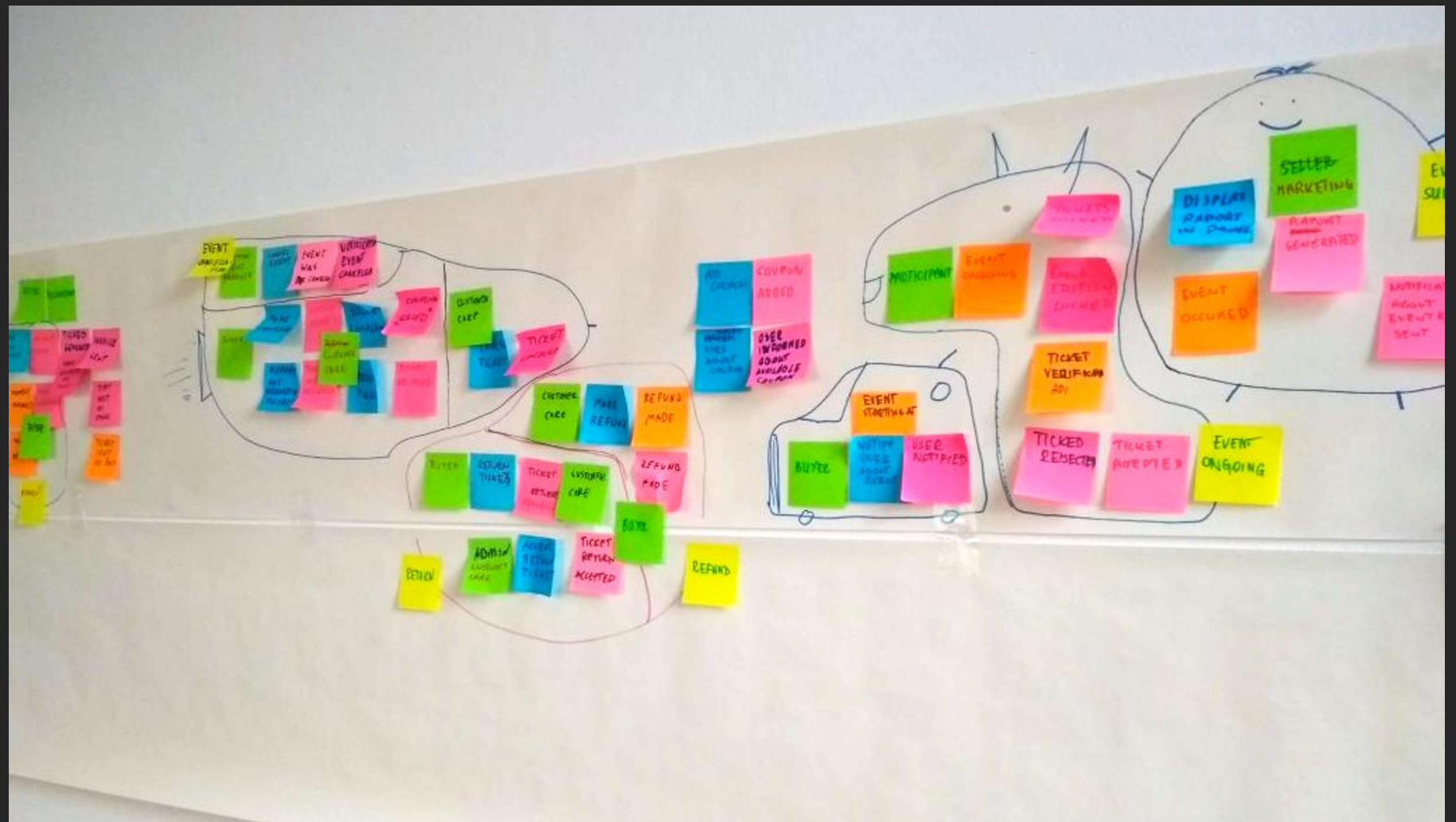
**Revenons  
au code !**





# **Identification des différents métiers/domaines Aka Bounded Contexts**

# Event Storming workshops





Réservations  
des places

Disposition  
des sièges

Marketing

Planification des  
représentations

Tarification

Impression  
des tickets



Réservat**ions**  
des places

Disposit**ion**  
des sièges

Market**ing**

Planificat**ion** des  
représentations

Tarificat**ion**

Impress**ion**  
des tickets

# Personae

Barnabé,  
actor

Julie,  
marketing

Eric, client

Direction

Terminal

# Synonymes

Réservations  
des places

Disposition  
des sièges

Marketing

Planification des  
représentations

Tarification

Impression  
des tickets



**@romeu@mastodon.social**

@malk\_zameth

In the culinary bounded context: 🍅 is a vegetable

In the botanic bounded context: 🍅 is a fruit

In 🍃🍷 bounded context: 🍅 is feedback

2:17 PM · 10 nov. 2017



# Tips identification BC

“/t/s/ion” sjõ  
“ing”

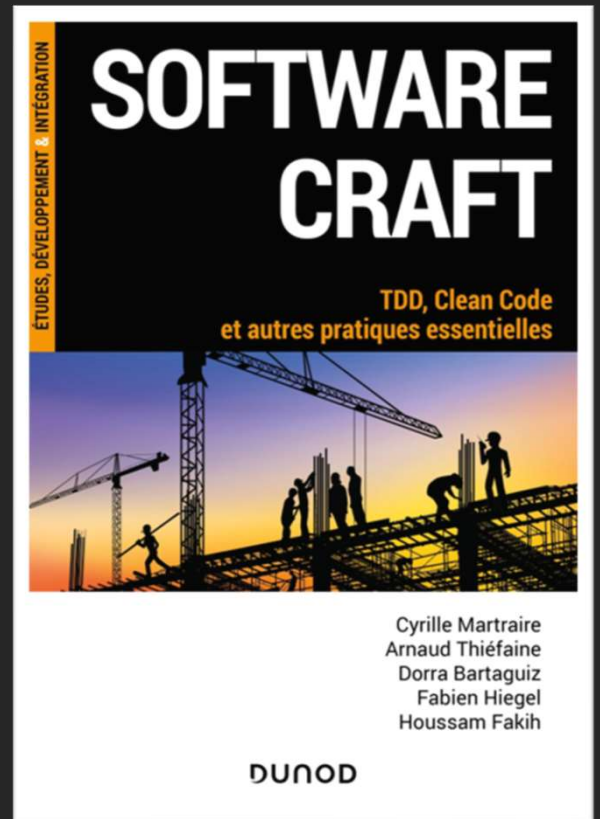
Différents personae

Synonymes

Termes à sens multiples

# <https://kahoot.it>

----



Rendons *DDD* aux devs !

---

# Résumons !



1<sup>e</sup> partie

Value Object vs Primitives

Séparation du métier de la technique

Bounded contexts







Devoxx France 2023

---

Rendons le  
DDD aux devs !  
2<sup>e</sup> partie

Arnaud THIEFAINE  
Dorra BARTAGUIZ

AROLLA

Rendons *DDD* aux devs !

## SUMMARY



1<sup>e</sup> partie

Kata de modélisation (Theater Kata)

Value Object au secours des primitive-obsessions

Séparation du métier de la technique

pattern sandwich

Identification des Bounded Contexts

2<sup>e</sup> partie

Séparation du métier et de la technique

archi hexa

Protéger les invariants (Aggregate, Entity, Value Object)

Service & repository

Découpage en Bounded Contexts

# Les 3 complexités du logiciel

Essentielle : nature du problème à résoudre  
Intrinsèque au métier

Incompressible



Obligatoire : l'outillage technique qui supporte la  
résolution du problème  
Liée aux contraintes d'exploitation

Compressible

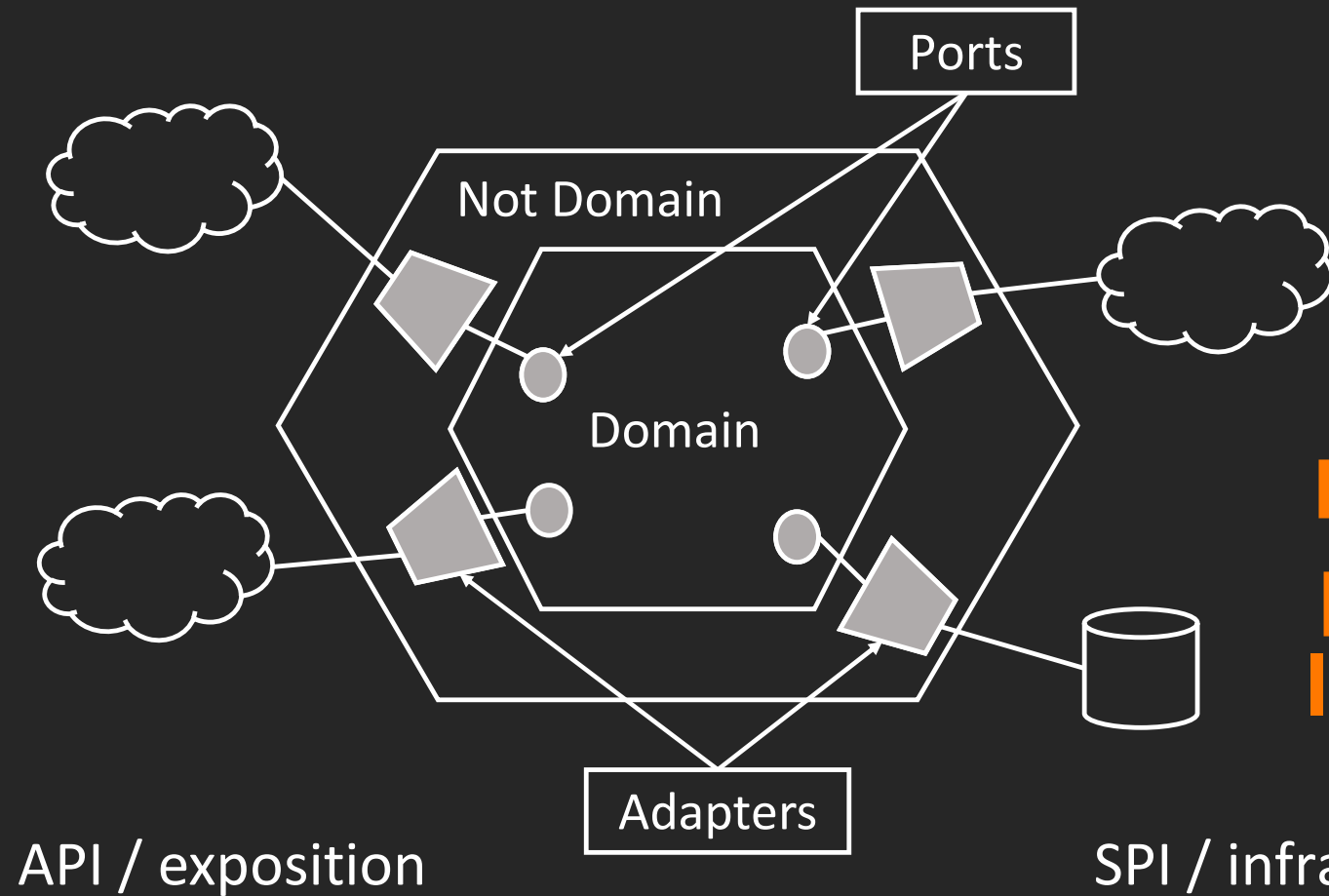


Accidentelle : les erreurs de développement  
L'essentiel de la dette technique

Suppressible

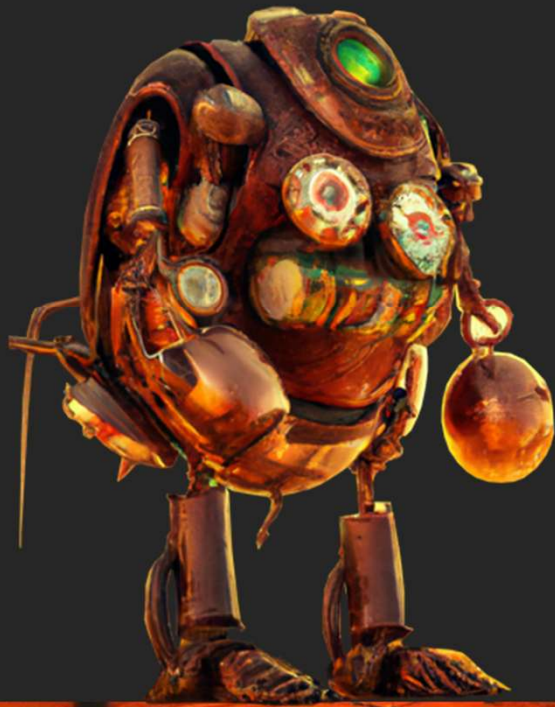


# Hexagonal architecture



**Mock infra  
plutôt que  
le domaine**





**Revenons  
au code !**

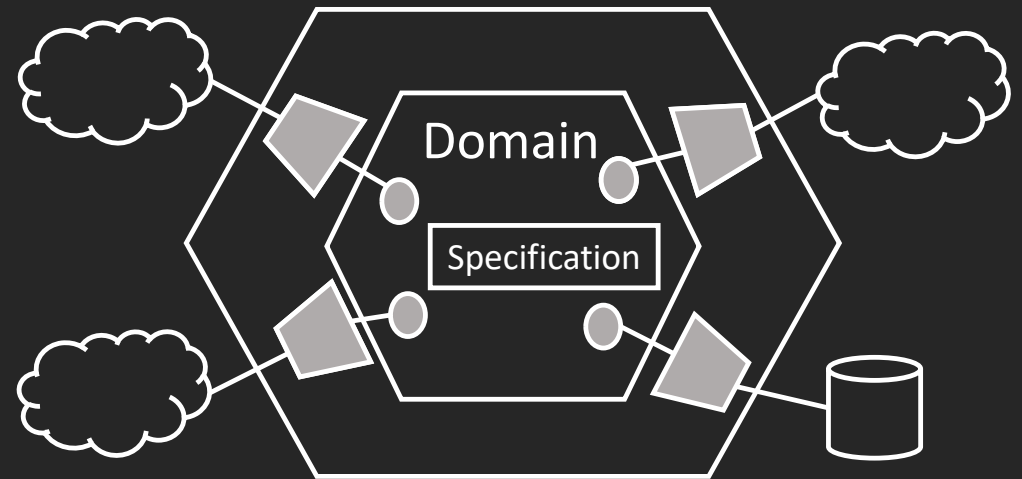
# Specification

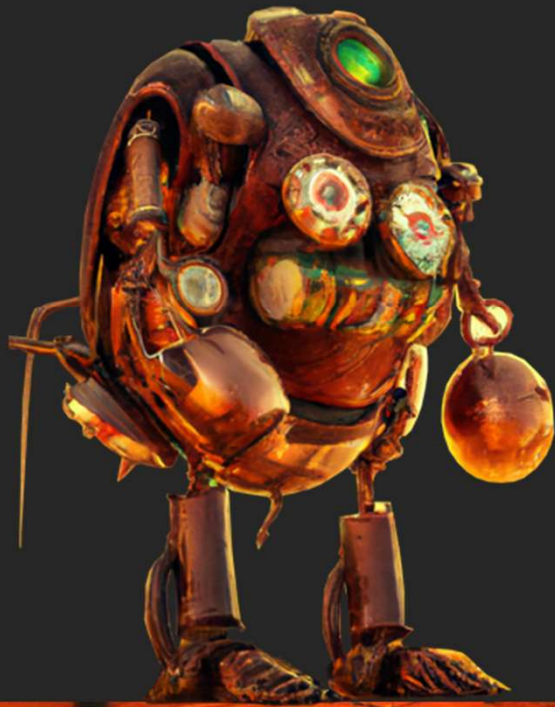
Invariant de domaine de type prédicat

Valide un objet domaine avec isSatisfiedBy()

Modélise une règle de gestion

Configurable





**Revenons  
au code !**

# Repository

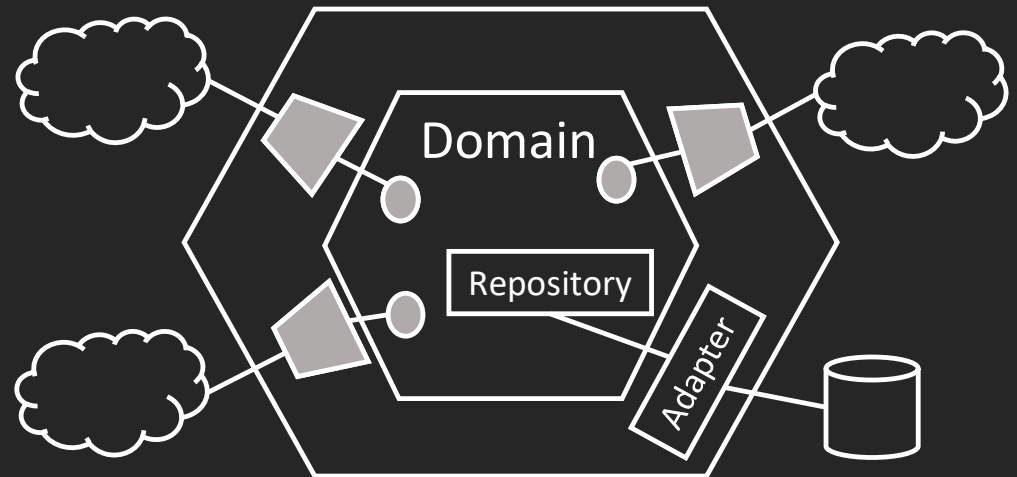
Collection-oriented

Manage transaction  
in repo not the service

Effet de bord

Injectable

Ubiquitous language  
pour le nommage





# Extract Repository

1. Move to a new target class (suffix Adapter)
2. Convert to instance method
3. Extract interface
4. Pull member up
5. Extract value as field initialized in constructor
6. Type migration of field to use interface
7. In constructor, extract parameter for the value
8. Type migration of parameter to use interface

# Aggregate

Identité unique pour la grappe

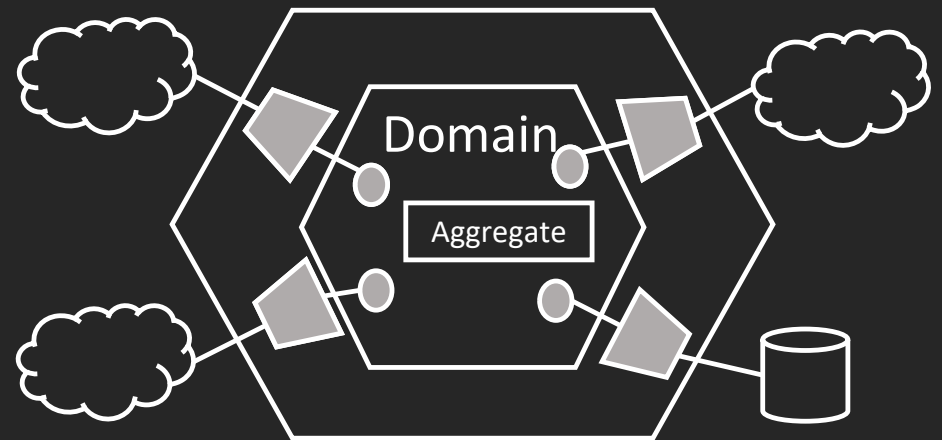
Aggregate Root

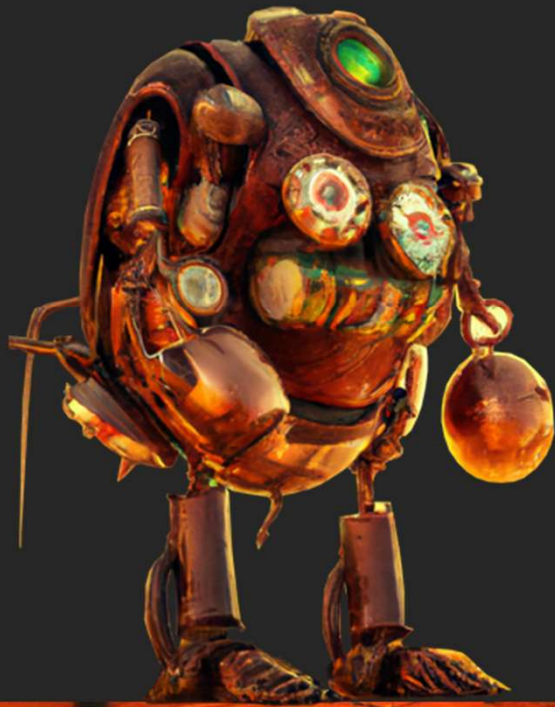
Invariant à protéger

Consistance

validité

transactionnelle (save/load)





**Revenons  
au code !**

Performance

Id

play

startDate

endDate

performanceNature

Topology

Reservation

Allocation





## Performance

Id  
play  
startDate  
endDate  
performanceNature

Topology

## TheaterSession

titre  
startDate

Reservation

## PerformanceNature

value

Allocation

# Value object

Pas d'identité

Egalité par valeur

Immuable

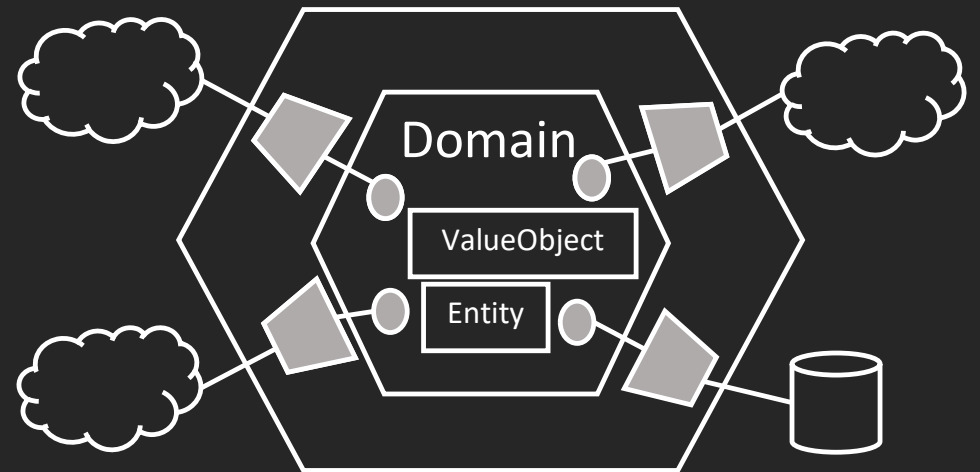
Constructeur paramétré  
Factory method

Side-effect-free

# Entity

Identité

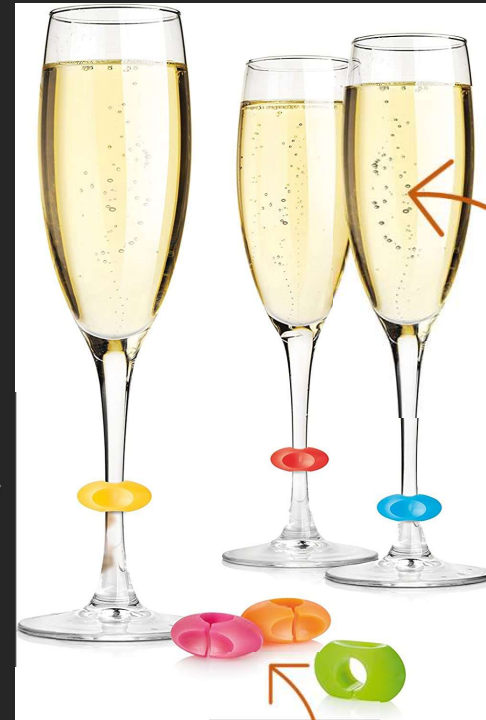
Continuité dans le temps



**Value  
object**

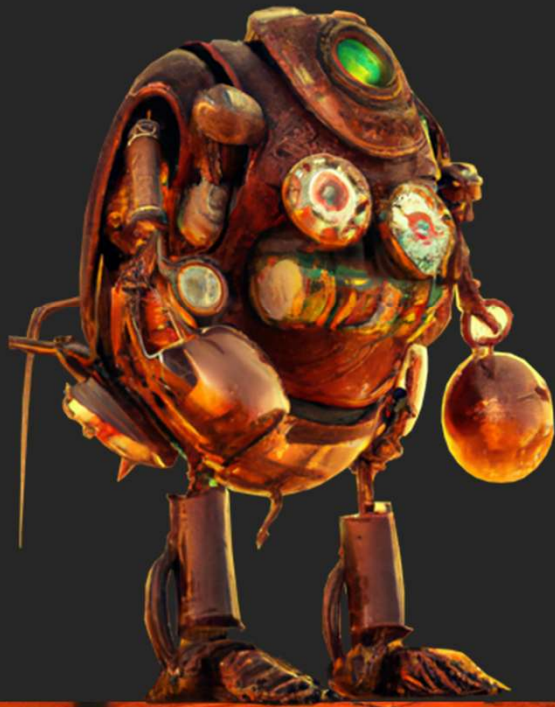


**Entity**



**Vide /  
rempli**

**Identity**



**Revenons  
au code !**



# Theater topology

Central zone  
(standard)



Row A

Row B

Row C

Row D

Row E



Row F

Row G

Balcony 1  
(premium)



Row I

Row J

Row K

Balcony 2  
(premium)

# Theater topology

Central zone  
(standard)



Row A

Row B

Row C

Row D

Eric needs to  
reserve 4 seats

Row E

Row F

Row G



Balcony 1  
(premium)



Balcony 2  
(premium)

Row I

Row J

Row K

# Theater topology

Central zone  
(standard)



Row A

Row B

Row C

Row D

Eric gets seats  
B3, B4, B5 and B6

Row E

Row F

Row G



Balcony 1  
(premium)



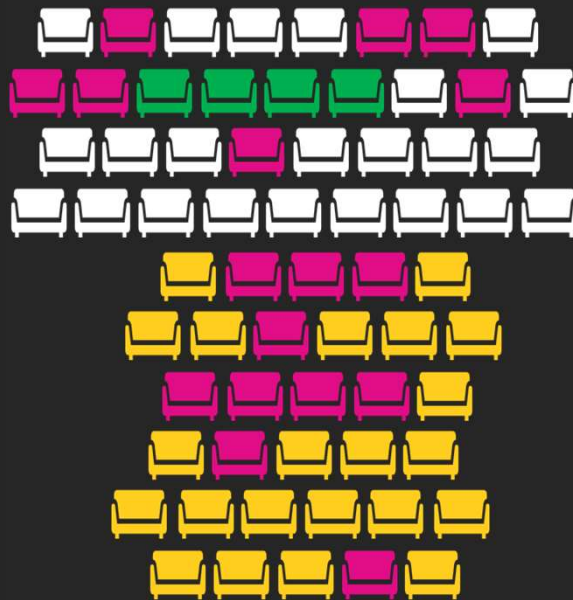
Balcony 2  
(premium)

Row I

Row J

Row K

# Seat allocation



Row A

Row B

Row C

Row D

Row E

Row F

Row G

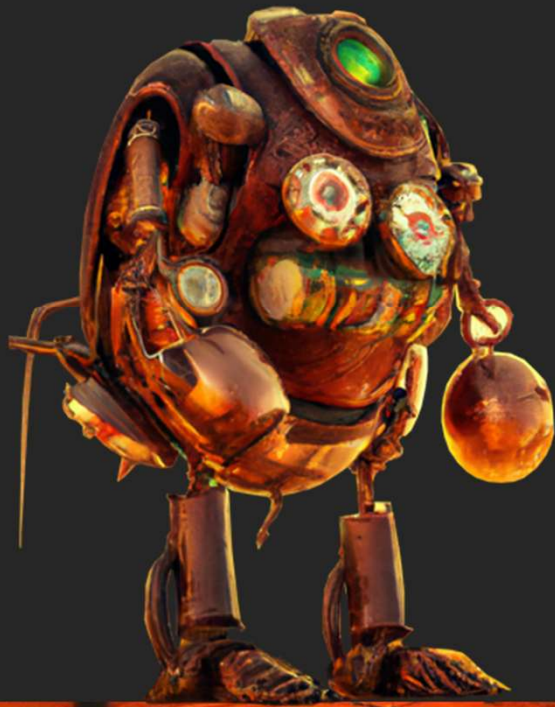
Row I

Row J

Row K

No more Zones





**Revenons  
au code !**

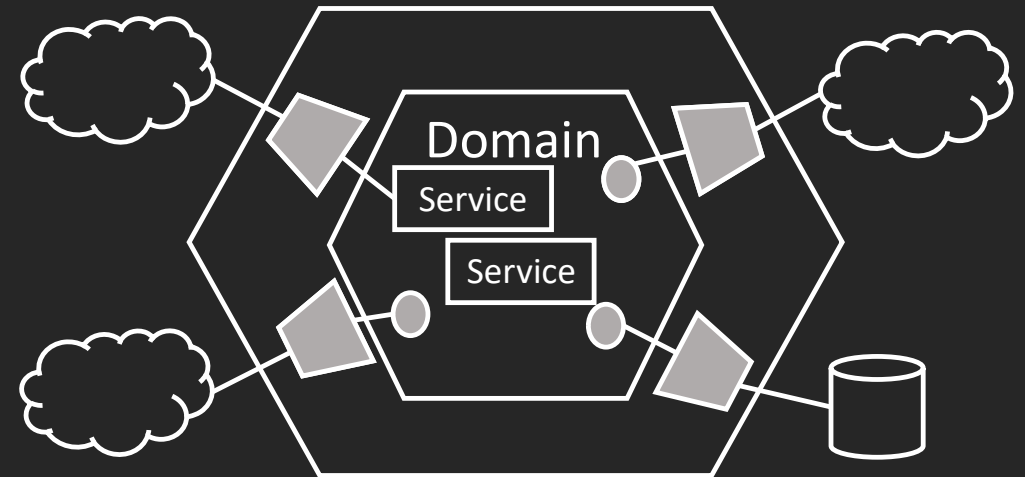
# Service


Stateless

Injectable

Logique métier  
ni dans VO  
ni dans Entity

Ubiquitous language  
pour le nommage

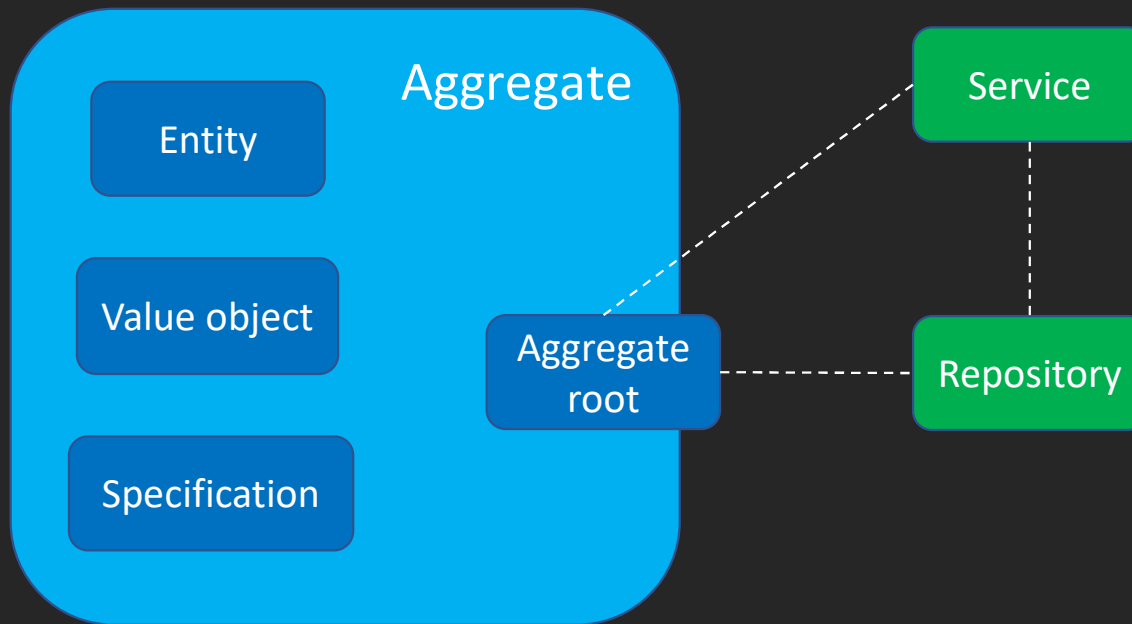




*Make sure **you need** a service. [...] Using Services overzealously will usually result in the negative consequences of creating an **Anemic Domain Model**.*

*- Vaughn Vernon  
Implementing Domain-Driven Design*

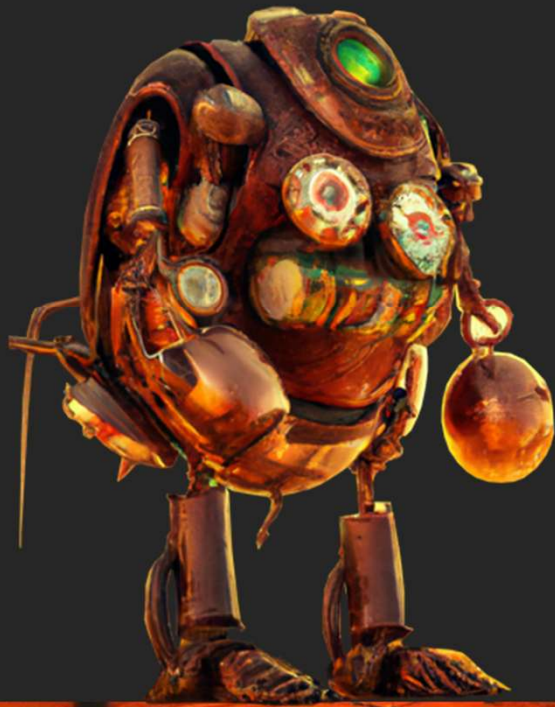
# Pour résumer



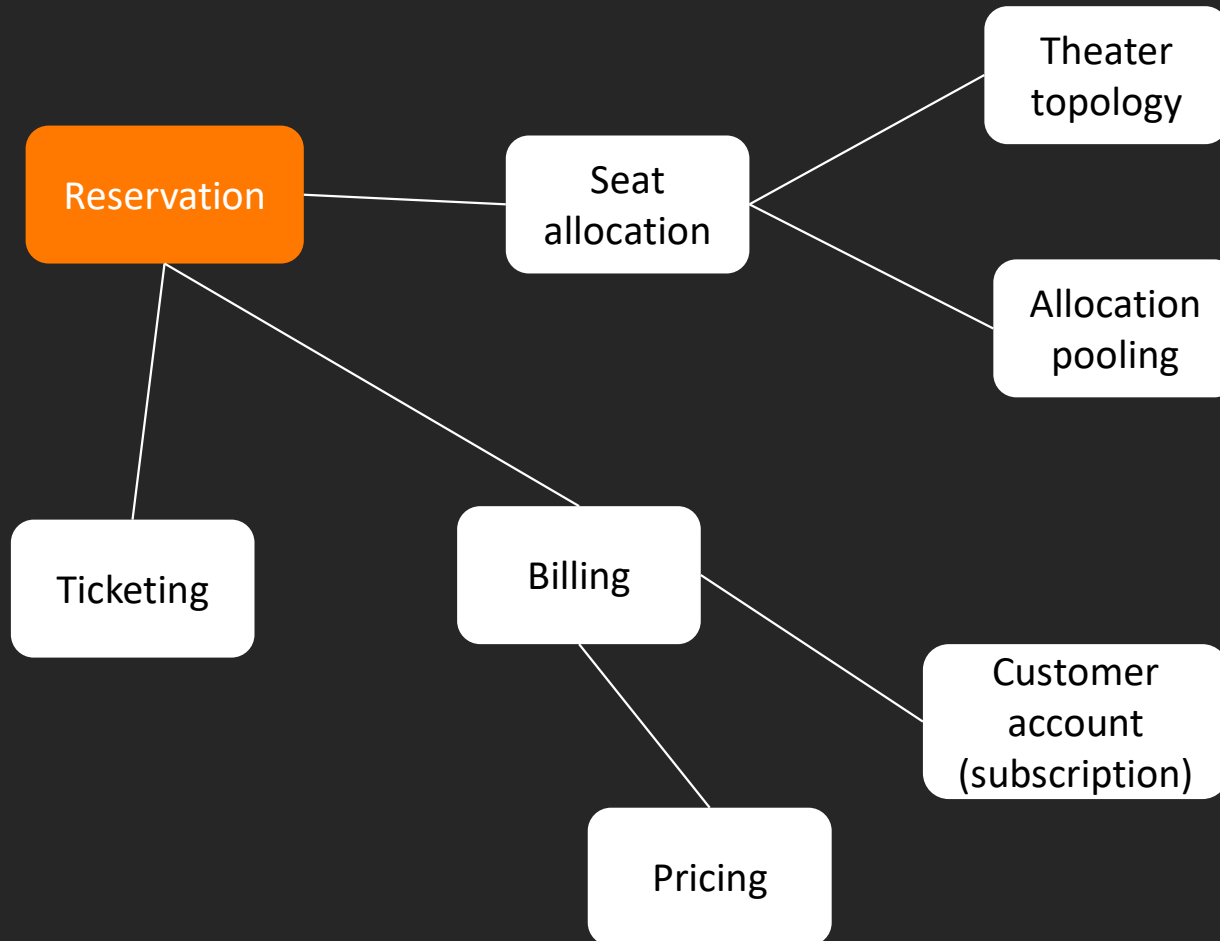
Modèle du domaine  
(instanciables)

Composants  
(injectables)



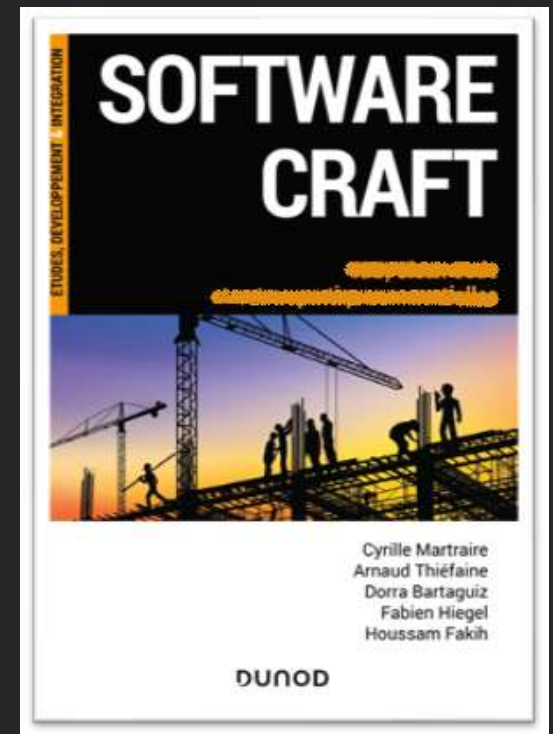


**Revenons  
au code !**



# <https://kahoot.it>

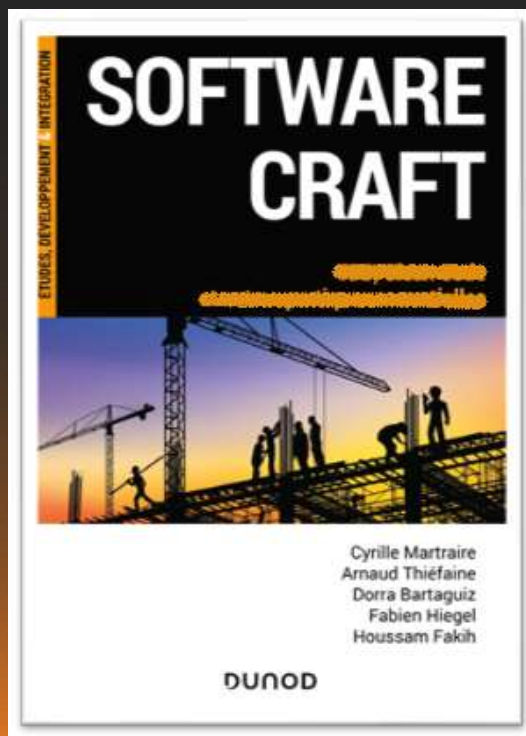
-----



# Rendons le DDD aux devs !



## À vous de jouer



Couverture de tests

Identification des Bounded Contexts

Refactoring en baby steps

Extract Value Object, Entities, Aggregates,  
Repositories, Services

Naming





THANKS  
FOR WATCHING