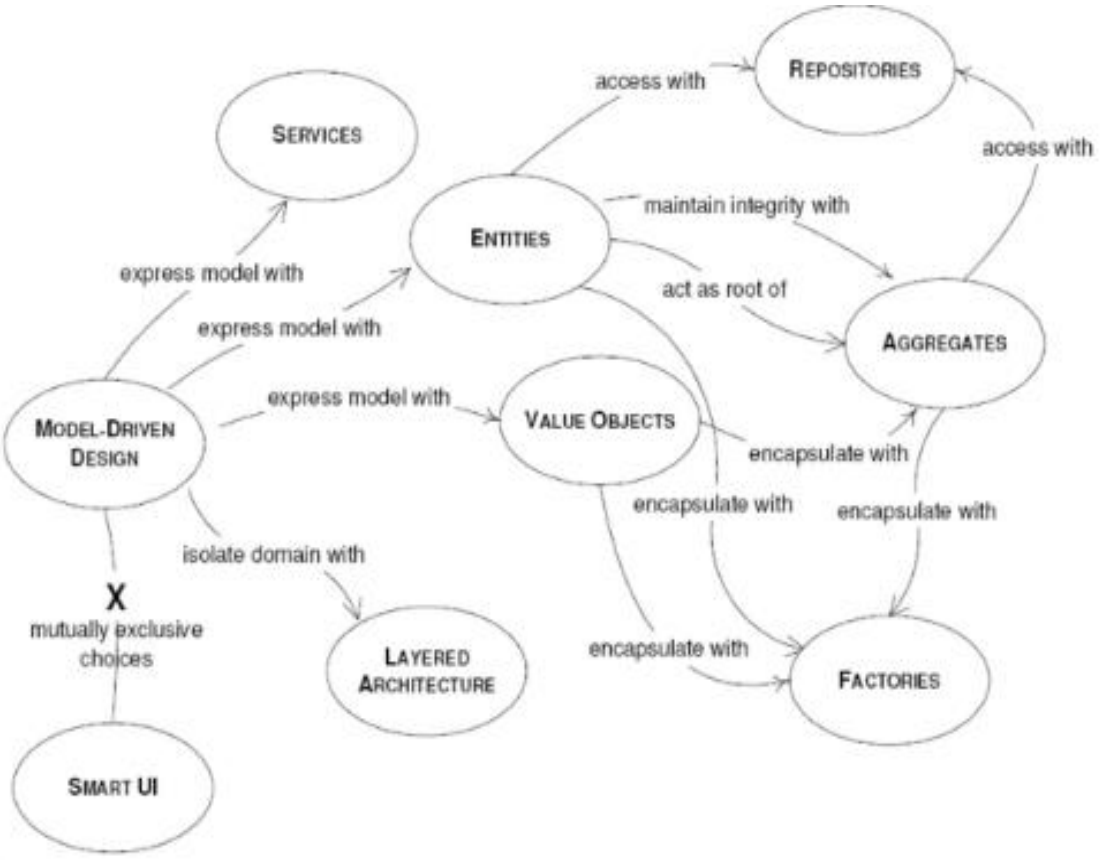
IACOB Iasmina , an IV , IS

TEMA 2 – Domain driven design – factory

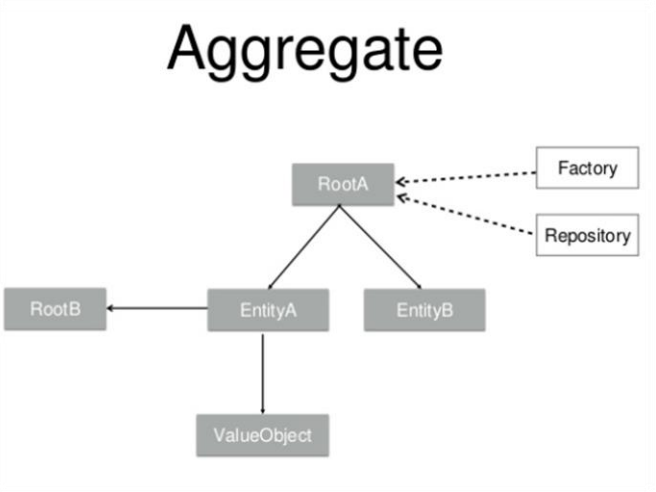
Domain Driven Design reprezinta un set de reguli pentru a realiza o dezvoltare rapidă a unei aplicației(o serie de principii de design) . Ajuta la dezvoltarea software pentru unele cerinte mai complexe prin conectarea implementării la un model în dezvoltare .



Factory reprezintă o parte importantă a domeniului unei aplicații ,ele fiind necesare pentru a menține obiectele de domeniu clare și pentru a le separa de logică irelevantă.

O fabrică este un obiect care are o singură responsabilitate, aceea de a crea alte obiecte. Fabrica este un obiect care produce agregate complicate sau, uneori, entități și obiecte valoare. Sunt adesea utilizate pentru a crea agregate. Agregatele oferă încapsulare și setează o limită consistentă în jurul unui grup de obiecte.

Atunci când se creează o metodă factory în agregate root, aceasta trebuie să fie una și nedivizată.Valorile obiectelor și entităților sunt create diferit. Deoarece valorile sunt nemodificabile, toate atributele trebuie transmise imediat după ce sunt create și se pot crea numai atribute specifice entităților.



Există trei tipuri de factory :

1. Factory method : este un model de design creator care oferă o interfață pentru crearea obiectelor în superclase, dar permite subclaselor să modifice tipul de obiecte care vor fi create.

2. Abstract factory class : folosită atunci când trebuie să determinăm de ce fel de obiect specific are nevoie clientul. Abstract factory class încapsulează atât logica deciziei, cât și logica creării.

3. Builder class : utilizată atunci când există o logică complexă necesară pentru a construi un obiect sau un întreg agregat. Aceasta tip este diferit de factory method prin aceea că obiectul fabricat nu aparține producătorului, iar diferența față de abstract factory este reprezentată de lipsa logicii pentru a decide ce clasă concretă să construiască.

Avantajele folosirii factory :

- Standardizează și oferă o modalitate de instanțiere a unui obiect astfel încât crearea de obiecte noi nu este pierdută odată cu cresterea aplicatiei.
- Realizează încapsularea cunoștințelor necesare pentru a crea un obiect complex sau un agregat.Fabrica trebuie să știe foarte multe despre structura internă și dependențele obiectului, însă fabrica va proteja această complexitate de lumea exterioară prin furnizarea unei interfețe care să reflecte obiectivele clientului și o vedere abstractă asupra obiectului creat.
- Furnizează un strat important de abstractizare care împiedică clientul de a fi dependent de o anumită clasă concretă. Este de datoria fabricii să instanțieze și să returneze o anumită clasă bazată pe un anumit tip.