# **Domain Driven Design – Domain Service**

## - Tema 2 -

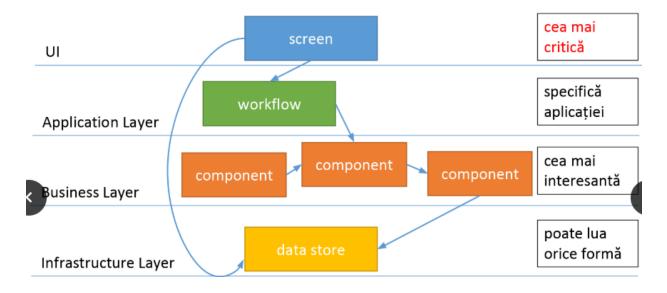
O cerință a unui client poate fi împărțită în mai multe zone de interes, extrăgându-se, astfel, temele relevante software-ului. Aceste zone de interes sunt descrise de către un domeniu, de exemplu, dacă zona de interes corespunde unei probleme de business, ea va fi descrisă de un domeniu de business.

Domain Driven Design sau DDD descrie problema prin separarea ei în subprobleme, în funcție de domeniul de interes. Aceasta se realizează prin extragerea conceptelor cheie ale subproblemelor și legându-le împreună. Astfel, DDD presupune colaborarea între dezvoltatorii și specialiștii domeniului, dar și dezvoltarea unui limbaj specific domeniului datorită complexității sale.

În alte cuvinte, design-ul bazat pe domeniu constă într-un set de modele pentru construirea de aplicații pentru întreprinderi. În domeniul software, aplicarea acestor idei împreună oferă posibilitatea construirii unor sisteme care îndeplinesc cerințele afacerii.

Acest concept a fost descris inițial de către Eric Evans în cartea sa *Domain-Driven DESIGN Tackling Complexity in the Heart of* Software. Eric sugerează, așadar, încă din titlul cărții faptul că atenția trebuie să fie asupra centrului aplicației. În perspectiva lui Eric, legătura dintre model și implementare este importantă deoarece aceasta face modelul să fie relevant prin asigurarea faptului că analiza care vine cu modelul se aplică produsului final, unui program rulabil. De asemenea, această legătură vine în ajutorul mentenanței și continuării development-ului deoarece codul poate fi interpretat prin înțelegerea modelului.

#### **Business Application model**



Spre deosebire de domeniul Event care definește un eveniment, ceva ce se va întâmpla, service domain este folosit atunci când o operație nu aparține nici unui obiect. Acesta reprezintă un proces al modelului domeniului care face referire la interacțiunea cu mai multe entități, desi nu face parte din nici o entitate. De asemenea, permite accesul la repository, acesta putând fi injectat, și pune la dispoziție operația fără stare coordonată de nivelul aplicație.

Application service se situează deasupra nivelului domeniu, astfel, nu poate fi chemat de entități din cadrul acestui nivel, spre deosebire de service domain, care poate apela și poate fi apelat de entități.

## Exemple:

Eric Evans a propus ca exemplu pentru domeniul service funcția de transfer între două conturi bancare deoarece implementează importante reguli de business și deoarece un transfer bancar este un termen semnificativ pentru bancă. În acest caz, funcția serviciului este de a cere celor două obiecte *conturi* să își îndeplinească sarcina. Din cauza faptului că sunt implicate două conturi și câteva reguli globale, se sugerează crearea unui obiect *transfer fonduri* care să reprezinte atât cele două intrări, cât și regulile și istoricul transferurilor.

Împărțirea pe layere a serviciilor unui transfer bancar este următoarea:

#### Fartitioning Services into Layers

Application	Funds Transfer App Service: 1.Digests input (e.g. XML request), 2.sends message to domain service for fulfillment, 3.listens for confirmation, 4.decides to send notification using infrastructure service.
Domain	Funds Transfer Domain Service: Interacts with necessary Account and Ledger objects, making appropriate debits and credits, supplies confirmation of result (transfer allowed or not, etc.)
Infrastructure	Send Notification Service: Sends emails, letters, etc. as directed by application.

Printre funcțiile ce pot fi implementate în domain service se pot număra funcțiile de trimitere a unui e-mail sau a unui mesaj text (SMS), funcția de conversie a unei entități Correspondence într-un PDF.

Sebastian BUDEA Ingineria Sistemelor Anul IV, grupa 1.2