Hopota Raul-Sorin

An IV, AC , IS

Tema 2 PSSC

Domain Driven Design- Aggregation Root

Înainte de a putea vorbi de Aggregation Root trebuie sa vorbim despre Domain Driven Design(DDD). In mare Domain Driven Design încearcă sa își îmbunătățească software-ul prin crearea unui model al unui proces sau sistem din lumea reala. In folosirea Domain Driven Design se lucrează împreună cu un expert din acel domeniu care poate explica cum ar trebui sa funcționeze sistemul in lumea reala. De exemplu, daca lucrezi la un sistem care se ocupa cu plasarea de pariuri pentru meciuri de fotbal atunci expertul tău poate fi un agent de pariuri experimentat. Per total ideea este ca programatorul ce se ocupa de partea de software trebuie sa descrie funcționalitatea sistemului pe care dorește sa îl dezvolte astfel încât acesta poate si citit si aprobat de către expert. Aceasta descriere este una conceptuala si se numește limbaj omniprezent (ubiquitous language) si pentru exemplul de mai sus ar conține definirea unor cuvinte cum ar fi : meci, pariu, cote si așa mai departe. Domain driven design oferă de fapt un ghid sau manual care descrie cum ar trebuie sa interacționeze obiectele tale si le împarte in categorii: value objects, entities si aggregation roots.

Aggregation roots sunt obiecte ce poseda alte obiecte. Este un principiu complex care se bazează pe ideea ca anumite obiecte nu au sens sau nu pot exista daca nu au un posesor. De exemplu, o mașină nu ar putea exista fără componentele sale: roti, motor, volan s.a.m.d. Așadar aceasta mașină este un aggregate ,un obiect format din alte obiecte, un obiect ce nu ar avea sens fără obiectele ce îl compun. Putem spune ca un aggregate este un ansamblu de obiecte pe care le tratam ca fiind unul singur si fiecare aggregate are o rădăcină(root) ce conține niște limitări ce ne spun ce este in interiorul si exteriorul acestui ansamblu de obiecte. Aceasta rădăcină este singurul obiect accesibil global in aplicația noastră.

Sa spunem ca vrei sa cumperi o masa dar in magazin ea este dezasamblata si toate componentele sunt într-o cutie si trebuie asamblate. Scoți afara toate componentele le pui una peste alta si nu obții o masa. Obții componentele din care e formata masa si tu trebuie sa le asamblezi după instrucțiuni. Masa reprezintă un aggregate, deci ansamblul de obiecte ținut laolaltă de aceste instrucțiuni/reguli pentru a se comporta ca un singur obiect. Asta reprezintă de fapt un aggregate : suma dintre obiectele componente si regulile de asamblare, ai nevoie de ambele pentru a construi masa.

Aggregate-ul este un model ce conține toate informațiile necesare de care avem nevoie pentru a schimba ceva. Informația este organizata in componente, care la rândul lor pot fi compuse din alte componente mai „mici”, si reguli ce trebuie respectate. In primul rând avem

nevoie de un „business case” care are rolul de a schimba ceva, de a putea face modificări pentru ca un aggregate trebuie sa poată schimba stări. Pentru exemplul cu masa de mai sus nu se vrea construirea unei mese dar asta trebuie sa facem pentru a obține o masa, deci acesta este „business case-ul nostru”: asamblează masa.

După identificarea business case-ului trebuie sa identificam in continuare ceva numit business concept, adică relația pe care aplicația noastră o are cu domeniul din viața reala. Dorim sa alegem un model al domeniului specific pentru business case-ul identificat mai sus. Pentru exemplul cu masa avem nevoie de o un model care ne spune care sunt părțile componente importante dar si care sunt regulile necesare asamblării mesei. Pentru acest caz regulile reprezintă instrucțiunile de asamblare iar componentele sunt piesele mesei dezasamblate.

Un aggregate definește limitele consistentei, adică tot ce este in interiorul acestui aggregate trebuie sa fie consistent imediat. Acest lucru este important pentru ca ne spune ca oricât de multe schimbări trebuie efectuate, trebuie sa le vedem pe toate ca o singura modificare mai mare formata din schimbări relaționate mai mici care trebuie sa reușească împreună. Având laolaltă toate aceste componente si reguli ne spune ca avem de-a face cu un grup, ansamblu de obiecte care se comporta ca o singura unitate care trebuie sa fie mereu consistenta. Deci la modelarea unui astfel de aggregate este foarte importanta prezenta si participarea unui expert din domeniul din care face parte aplicația.

Rolul unui aggregate este unul foarte specific. Avem nevoie de un model pentru ca avem nevoie de niște schimbări de stare valide. Scopul unui aggregate este acela de a controla schimbarea ci nu de a fi schimbarea. Nu suntem interesați de starea propriu-zisa, suntem interesați in garantarea ca schimbările pe care dorim sa le efectuam respecta regulile si de aceea folosim modul de gândire din domeniul din viața reala, privim lucrurile ca si cum am fi o parte din acest domeniu. O instanța a unui aggregate comunica faptul ca totul este pregătit si in regula pentru ca o schimbare sa se întâmple.

O schimbare e văzută ca unul sau mai multe evenimente ale domeniului care sunt generate de aggregate. Aceste evenimente trebuie persistate si aplicate. Putem spune ca menirea unui aggregate este aceasta: in funcție de input-ul primit si regulile ce trebuie respectate, următoarele schimbări au loc: X se întâmplă cu aceste detalii. Programatorul poate face ce dorește cu aceste schimbări pentru ca treaba aggregat-ului se termina acolo.

Rolul unui aggregation root este acela de a impune regulile/limitele de consistenta. Deci rolul unui aggregation root poate fi împlinit de către un obiect sau chiar o funcție. Practic aggregation root-ul se asigura ca componentele necesare exista si ca regulile sunt respectate, pentru exemplul cu masa aggregation root-ul este persoana care asamblează masa.