What is the relationship between DDD and the “Onion Architecture”?

Asked 9 years, 7 months ago Active 1 year, 5 months ago Viewed 6k times

36

2

What is the relationship between Domain-driven design (DDD) and "The Onion Architecture" of Jeffrey Palermo?

c# java architecture domain-driven-design onion-architecture

edited Mar 21 '14 at 17:59

asked Aug 3 '10 at 18:29
ulrichb
17.2k 6 65 83

3 Answers

27

In my opinion - they complement each other - but from very different perspectives.

Onion Architecture is all about making the Domain/BusinessLogic independant on 'inferior' things like data-acccess, UI, services etc. The Onion Architecture doesn't really care how you made the domain you have - it's adamant about protecting it from outside dependencies.

Domain Driven Design is all about how you model your Domain and what you call your objects. Meaning that each Domain class should have a direct relation to what it represents in the business domain it is adressing (ie. the physical/real world). So a Customer object should be named a Customer in code - it should have the same rules as a Customer does in the real world (or as close as it is possible).

edited Sep 10 '18 at 18:28
UuDdLrLrSs
5,594 5 30 46

answered Aug 3 '10 at 18:46
Goblin
7,212 3 30 34

If you look at the image that describes the onion architecture in the link you provided, the *Domain Model* layer is what DDD focuses on.

30

Onion is an architectural pattern for a system, whereas DDD is a way to design a subset of the objects in the system. The two can exist without eachother, so neither is a subset of the other. If you were to use them together - then as a whole the part that is designed using DDD would be a subset of the entire system.

To use a (probably bad) analogy: Onion is a pattern to design a house, and DDD is a way to mill the wood that is a part of the house.

edited Aug 3 '10 at 19:35

answered Aug 3 '10 at 18:46
Steven Evers
14.8k 16 70 117

0

I think the two differs each other on the "how you design and what is your overall philosophy" on the system itself.

With the onion architecture all the world is around your Data store. This means that you do not care on how you "do the job" but you care more on "i want the job done". Obviously best practises are (small methods, good variable names, maybe some design patterns like Sigleton etc. etc) are still have a meaning in this scope of view but more for the code itself and not so more for the whole application overview.

On the other hand when we talk about DDD we talk about Business and when we take a decision about the architecture we always have the Business model we want to solve on our minds. And when I say decision I mean the total from decisions a system may have (from Variable/Class/Function names, to where we place a part of code and at the end of the day how we talk about this system itself). We could say that a DDD is a "programming abstraction" of the Business Model). But of course we can not make DDD with out have a business to solve e.x we can not make DDD on a program like Shazam but we can make DDD on a program like Facebook or Spotify.

My option is that you can not mix each other but instead it's the first decision.

answered Mar 8 '18 at 12:05
dios231
624 7 18