**Pattern, Principles, and Practices of Domain-Driven Design(Scott Millett with Nick Tune)**

**Individual Volume 1 Summary Edition**

**By Hussein Nakhostin Saed**

**On Friday, April 5, 2024**

**On Friday, Farvardin 17, 1403**

**Chapter 4**

**Model-Driven Design**

**From Page 41 up tp 58**

Domain Driven Design needs to produce a model to serves analysis model for business people which is implemented using same terminology and concept in code this proccess known as model driven design.

Domain is our problem area that we are working within but domain model is an abstraction that represent a view of problem domain and complex logic and policies shown and intent is to solve business problems model generated by collaboration between development and business teams.

Domain model is in center of Domain Driven Design that represent a shared language between development and bussiness teams, it prepares a view to describe problem domain which designed only to meet the needs of business use cases.

Analysis model known as business model or UML is collection of artifacts which describe model and is an blueprint for development and business team to solve problem domain.

Development and business teams obey a common and global language to named UL and analysis model that described via this language.

Analysis implementation accor in a synergy which stablize and sync UML and implementation.In summary, at first development and business team must understand the business, after implementation, some change that does not fit development.this time they should talk to business experts if facing a new challenges to solve them, otherwise, analysis and code model doesn`t sync together. This is upfront design.

Any change that accur during development in model must pass a loop of synergy which include development and business team feedback together this is team modeling.

In UML, developers should seperate tecknical concerns from analysis model, on the other hand, business experts should pay attention to buiness process, not to business conclusion and tecknikal process.

For shaping the UL we have to comply some rules like create a glossary and a terminology contract for UML to avoid confusing and prevent from any tecknical names like design pattern names and naming exceptions and .....

Listen carefully to conversetion about business process and focus on it to create some scenarios which make your model invalid.we can pass using specification pattern, in this pattern, specification classes are scenarios to validate the model and process.

The best way to create effective domain is to firstly focus on areas of the application that are important to the business and ignore the parts like manage data and CRUD management.

Abstraction come at a cost. So, do not seek to abstract every domain concern.remember it is always better to be explicit rather than hiding an important domain concept behind layers of needless abstraction.but when should we abstract? Take the example of traveling to work.the abstract concept would be to commute whereas walking, talking the train, or driving is a variation of that concept;i.e..., the concrete implementation.if there were no variation in traveling to work we would not need to introduce an abstraction concept such as commuting.according to my says abstraction comes at a cost. So, reader sould achieve concept by reading abstraction without drill down in to the details.

Create abstraction in low level will cuase a greate of friction when you nedd to refactor a model.simple problems do not require complex solutions and do not need to create UL,focus your efforts with domain experts on the complex or important core domain.

Always challenge tourself and ask the questions,”Am I working within the core subdomain? Does this problem require a rich domain? Does the business care about this area of the application? Will it make a difference? Is it important to the business and do they have high expectations of it or do they just want it to work?”.