**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 7, 2024**

**On Friday, Farvardin 19, 1403**

**Chapter 5**

**Domain Model Implementation Patterns**

**From Page 59 up tp 72**

The focus of DDD is to manage complexity. There are some patterns to implement models in code , on the other hand there is no best practices to selecting pattern to concrete domain logic. As long as you isolate domain logic from technical concerns, you can implement MDD and hence DDD.

Domain layer is at heart of application. It isolate complexities of somain model from the accidental technical complexities. In modt cases, the domain layer makes up only a small part of your application.

There are various pattern to implement model like domain mode, transaction script, table module, but, there is a hint in here, large system does not implement in same way that means, maybe a part of system or same models are important and some one is less than, thats why, we should choose different pattern for multiple part of system. As you read, we wrote three patterns which they presented in the book “Pattern Of Enterprise Application Architecture” by Martin Fowler. There is some other pattern like active record pattern, anemic domain model pattern, functional pattern.

Domain Model Pattern:

The Domain Mode Pattern is an object-oriented model that incorporates both behaivor and data, also business process and association, rules and rich domain logic. In this pattern we considor that there is no database and using Plain Old C# Object (POCO) design our domain objects and cover completely Persistence-Ignorant Principle this ability to focus only on the domain model enables the disgn of the domain logic to be driven by the abstraction of domain hence DDD. You should focus on the verbs and actions of a problem domain. When modeling as this will help you concentrate on behaivor rather than state. In domain model each object is responsible for a specific task. The domain model pattern can be costly to implement and we use this to manage complex logic. So, trying to model everything and apply object-oriented practices would be a waste of effort that would be better spent on your core domain.

If application does not have frequently changing logic, it`s a useless try to apply the domain model pattern.

Transaction Script Pattern:

Follow a procedural style of development rather than an object-oriented approach. Asingle procedure is create for each of business transaction and grouped in some kind of static manager or service class.

One of the strength of transaction script pattern is easy to learn and to implement. All logic for an operation is contained within a single service method. It is usefull for the part of your domain that have little or no logic. If apply this pattern to complex logic, this pattern become hard to manage becuase duplication accur quickly.