What is Domain-Driven Design (DDD)

A definition of DDD as a software design discipline

By Mathias Verraes
Updates & discussion:
https://verraes.net/2021/09/what-is-domain-driven-design-ddd/

Domain-Driven Design is a software design discipline centred on the principles that:

- Software for a complex domain requires all designers (engineers, testers, analysts, ...) to have a deep, **shared understanding of the domain, guided by domain experts**
- That understanding is **rooted in language**: the domain language should be formalised into a Ubiquitous Language (shared, agreed upon, unambiguous)
- The understanding is **expressed in a model**, shared between experts and designers, which express the problem space (as opposed to the solution space)
- The model must not shy away from **explicitly expressing the essential complexity** of the domain
- A complex domain can not be efficiently expressed as a single universal model and language, and must therefore be **separated into Bounded Contexts** (ie. an internally consistent language and model) by the system designers
- The model and language should be **in sync** with our understanding of, and changes in the domain, through **continuous refactoring towards deeper insight**

As a mnemonic:

DDD is a design discipline where you

- **Grasp the domain**
- **Agree on a language**
- **Express it in shared models**
- **Embrace complexity**
- **Separate models in contexts**
- **... and evolve them continuously**