Mastering the Basics of DDD

Domain-Driven Design with Java

Otávio Santana, Founder
otavio@os.expert

Technology as strategic resource

Everything is around software!

“Every business is a software business” CMMI

“Every Company is a Data Company” CIO Network

“Every Company is a Software Company” Forbes

Software Development

Failure?

- Hyper-Focused Planning And Design
- Unclear Or Undefined Client Expectations
- Unexpected Complexities
- Poor Collaboration Between The Product And Engineering Teams

INNOVATION

16 Obstacles To A Successful Software Project (And How To Avoid Them)

Forbes

Expert Panel® Forbes Councils Member
Forbes Technology Council COUNCIL POST | Membership (Fee-Based)

The Complexity Paradox

The More Answers We Find, the More Questions We Have

- Developer experience is a market
- Trade-offs
- The hype effect

Complexity is killing software developers

The growing complexity of modern software systems is slowly killing software developers. How can you regain control, without losing out on the best these technologies have to offer?

By Scott Carey
Managing Editor, News, InfoWorld | NOV 1, 2021 3:00 AM PDT

Evolutionary Software?

No ready to change anything!

"To prepare to write their classic text **Structure Design**, Ed Yourdon and Larry Constantine examined programs to find out what made them so **expensive**. They noticed that the expensive programs all had one property in common: **changing one element required changing other elements**."

Software Erosion

Continuous development

"Also known as software rot, code rot, software decay, or software entropy, software rot is either a **slow deterioration** of software quality **over time** or its **diminishing responsiveness** that will eventually lead to software **becoming faulty**."

Software Engineering?

A super inefficient way to build something

We waste time in meetings

Create complexity

Not ready for change

Premature legacy

No achieve client goals

Software Engineer

Definitions?

"Software engineering is the application of an empirical, scientific approach to **finding efficient, economical solutions** to practical problems in **software**."

Domain Driven Design

Designing software to meet business goals with less rework

Domain-Dreriven Design

Otávio Santana
@otaviojava

DDD?

The biggest misunderstanding in the software industry!

It is a framework!

Repository, Entity

Java

Microservices

Code only

Complex

DDD!

Handling and focus to the Business

Domains

Transfer Knowledge

Tactics

Strategy

Strategic DDD

The biggest mistake when implementing DDD

Domains and Subdomains

Context Mapping

Ubiquitous language

Bounded Contexts

Domains

Central area of a company's operations

Subdomain Type	Role	Business Differentiation	Complexity of Business Logic
Core Subdomain	Unique to the company, defining its identity and competitive advantage	High	High
Generic Subdomain	Common across all companies, standard business activities	Low	High
Supporting Subdomain	Supports core business activities without providing direct competitive advantage	Medium	Varies

Domains

The subdomains types

Otávio Santana
@otaviojava

Ubiquitous Language

The core's communication

Define common terminology

The same word can vary from context

The common language between experts and the engineering team.

Otávio Santana
@otaviojava

Bounded Contexts

Subset of the ubiquitous language

Otávio Santana
@otaviojava

Context Mapping

The relationships between these contexts

Shared Kernel

Customer-Supplier

Conformist

Open Host Service

Anticorruption Layer

Published Language

Separate Ways

Shared Kernel

A Common Ground

Two contexts share a part of the model, requiring close coordination and trust between teams.

Customer–Supplier

Collaborative Influence

The downstream (customer) depends on the upstream (supplier) and can influence its priorities or features.

Conformist

Forced Alignment

The downstream has no influence and must fully adapt to the upstream model as it is.

Anticorruption Layer

Protecting the Model

The downstream isolates its domain by translating the upstream model, preventing external concepts from leaking in.

Published Language

A Contract for All

Teams agree on a shared, well-defined communication language or schema to integrate smoothly.

Otávio Santana
@otaviojava

Published Language

A Contract for All

Teams agree on a shared, well-defined communication language or schema to integrate smoothly.

Otávio Santana
@otaviojava

Separate Ways

Independent Paths

Contexts remain isolated, evolving independently without integration.

Open Host Service

A Gateway to Services

The upstream provides a well-defined API or protocol that multiple downstreams can consume without tight coupling.

Otávio Santana
@otaviojava

Domain Storytelling

A Collaborative, and Visual Way to Build DDD

Otávio Santana
@otaviojava

Tactic DDD

The second step

Entity

Service

Repository

Aggregates

Value Object

Factories

Events

Entity

Identity Matters

An object defined by a unique identity that persists over time, even if its attributes change.

Value Object

Essence over Identity

An immutable object defined solely by its attributes; equality is based on value rather than identity.

Aggregate

Consistency Boundary

A cluster of entities and value objects treated as a single unit, with a root entity ensuring business rules and consistency.

Otávio Santana
@otaviojava

Repository

Gateway to Aggregates

An abstraction that provides controlled access to aggregates, handling persistence while keeping the domain pure.

Factory

Controlled Creation

A mechanism that encapsulates complex creation logic, ensuring invariants are satisfied when building aggregates or entities.

Domain Service

Logic Without a Home

Encapsulates domain behavior that does not naturally belong to an entity or value object, but is still part of the core domain.

Domain Event

Something Happened

Represents a significant occurrence in the domain, capturing a past fact and enabling communication across the system.

Application Service

Orchestrating Use Cases

Coordinates application workflows by invoking domain objects and repositories, ensuring the right sequence of operations.

Strategic and Tactical

Focus on tactics is the start of a huge mistake

Get something done
Working code isn't enough
Tactical tornado

Architecture DDD

Using Layers to abstraction

Otávio Santana
@otaviojava

Architecture vs Design

The boundaries to DDD

Architecture

The laws

- Everything in software architecture is a trade-off
- Why is more important than how

The Architecture styles

There is a world beyond microservices and Hexagon model

CQRS Architecture

Layered (n-tier) Architecture

Microkernel Architecture

Micro service Architecture

Orchestration Architecture

Spaced- Based Architecture

MVP Architecture

Event-Driven Architecture

DDD Architecture

Otávio Santana
@otaviojava

Governance

Automating processes and good practice rules

Using Annotations

Getting architecture evidences

```
@Entity
public class CreditCard {

    @Identity
    private BigInteger id;

    private String number;

    private String name;

    private YearMonth expiry;
}
```

Governance

Automating processes and good practice rules

```
@AnalyzeClasses(packages = "expert.os.examples")
public class IntegrationSampleTest {


    @ArchTest
    private ArchRule dddRules = JMoleculesDddRules.all();

    @ArchTest
    private ArchRule layering = JMoleculesArchitectureRules.ensureLayering();

}
```

ArchUnit

Putting test at your Architecture and Design

Otávio Santana
@otaviojava

xMolecules

Architectural abstractions in code.

"There is always a leak between the language and the architecture you want to express. Language, like any other tool, shapes what you can do with it, and sometimes it doesn't allow you to express things the way you'd want from an architectural standpoint."

Architecturally Evident Applications – How to Bridge the Model-Code Gap?

https://xmolecules.org/aea-paper.pdf

Otávio Santana
@otaviojava

Show me the code

Let’s show the code structure

```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
        int sum = 4 + 5;
        System.out.println("Sum: " + sum);
    }
}
```

Otávio Santana
@otaviojava

Software Engineering

The principle of the ultimate sophistication Engineer

- Documentation
- Persistence
- Code Design

- Software Architecture
- Leadership
- Test

Otávio Santana
@otaviojava

Expert

Otavio Santana

Software Engineer & Architect

- Java Champion, Oracle ACE
- JCP-EC-EG-EGL
- Apache and Eclipse Committer
- Jakarta EE and MicroProfile
- Duke Choice Award
- JCP Award
- Book and blog writer

Otávio Santana
@otaviojava

Domain-Driven Design

DDD Book

https://bpbonline.com/products/domain-driven-design-with-java

Otávio Santana
@otaviojava

Thank you!

Otávio Santana

OS Expert Founder, Software Engineer & Architect

otavio@os.expert

https://twitter.com/otaviojava

https://www.linkedin.com/in/otaviojava/

https://www.youtube.com/@otaviojava