Tackling Business Complexity in Enterprise Application's Development

Author: Renê Eduardo Pereira Cardozo
Supervisor: Prof. Dr. Daniel Macêdo Batista
Department of Computer Science — University of São Paulo

Introduction

The art of doing business requires constant negotiation and adjustments to cope with the different needs of clients.

These needs require flexility that leads to rules that are not so clearly defined and could be full of exceptions.

What are Enterprise Applications?

Martin Fowler, in Patterns of Enterprise Application Architecture, describes Enterprise Applications as:

- *"Enterprise applications are about the display, manipulation, and storage of large amounts of often complex data and the support or automation of business processes with that data".*

Examples of Enterprise Applications are:

- Enterprise Resource Planning (ERP)
- Customer Relationship Management (CRM)
- Payroll Management Systems and Financial Systems
- Mail Systems

Making Sense of Complexity

The Cynefin Framework was created by Dave Snowden at IBM to help leaders understand the context in which they are inserted.

The knowledge obtained from this analysis helps to establish strategies to deal with problems in different realities.

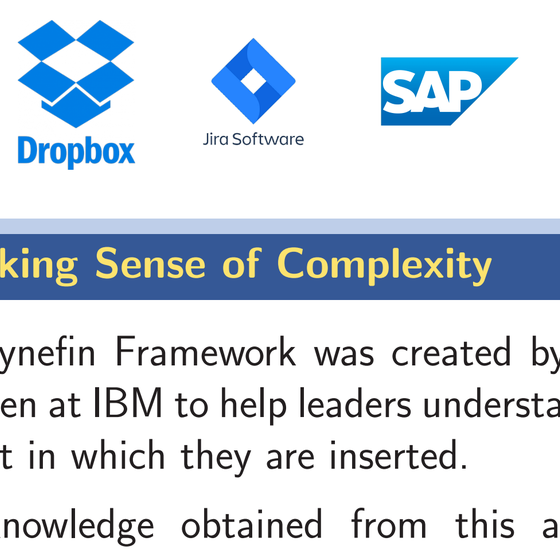

Figure: The Cynefin Framework

Objectives

Enterprise businesses are inherently complex. They are built aiming to support a myriad of requirements that arise from a business activity.

The objective of this work is to aggregate different techniques to manage complexity in those scenarios, especially involving domain modeling and patterns of Domain Driven Design (DDD).

The programming world is ordered, so to treat a complex problem, we need to be able to extract its relevant parts to the complicated or simple Cynefin's realms to establish a model.

Knowledge Crunching

To derive a satisfactory model, constant collaboration between domain experts and technical staff is required. Workshops can be used to extract the main requirements of the system.

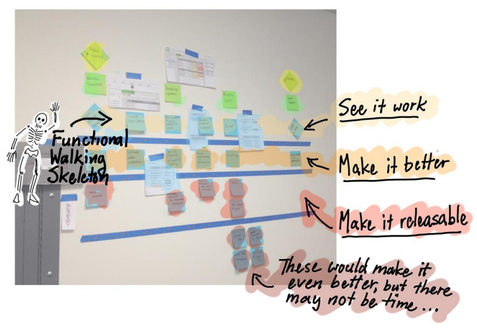

Figure: User Story Mapping Workshop

Figure: Event Storming Canvas

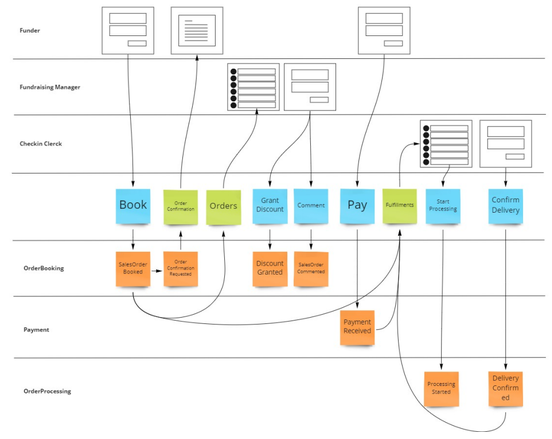

Figure: Event Modeling Online Canvas

Through this analysis it is possible to recognize bounded contexts:

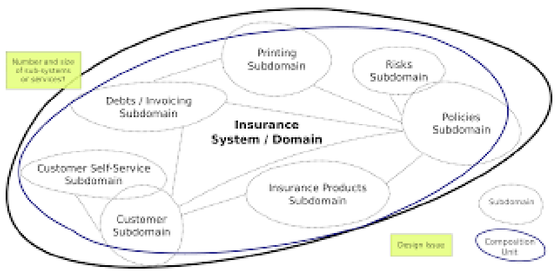

Figure: Bounded Contexts of an Insurance System

DDD Patterns

Aggregates are clusters of domain objects that establish a single consistent unit of data storage.

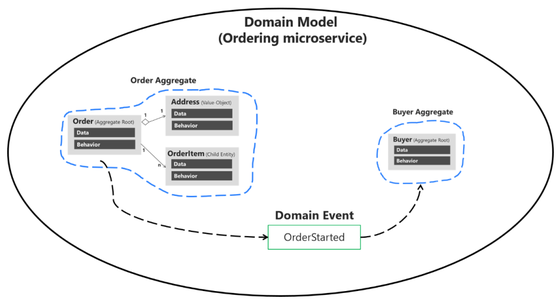

Figure: Aggregates are Always in a consistent state

Domain events are published by Aggregate methods and delivered to subscribers that can react to those events.

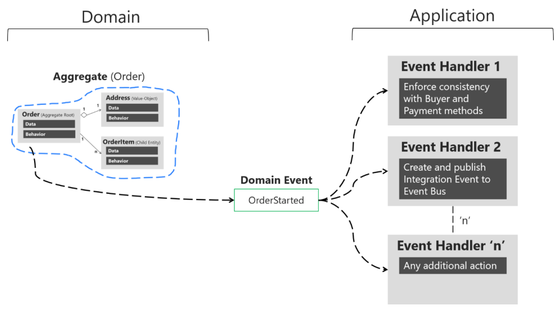

Figure: Domain Events from a Change to an Aggregate

Domain events allow the state of a business entity to be persisted as a sequence of state-changing events.

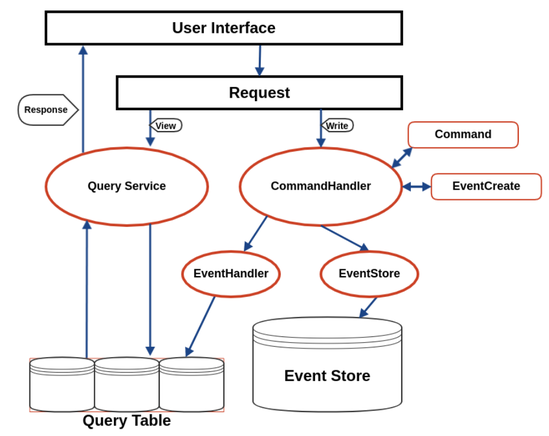

Figure: Event Sourcing and CQRS Basic Flow

Conclusions

To describe the complex reality of enterprise business as a software system it is necessary to establish a domain model through constant collaboration between domain experts and the technical staff.

Patterns from DDD and the use of an architecture based on Domain Events provide a loosely-coupled system that can be programmed reactively.

Those characteristics enable a better handling of the transient nature of business, real-time operations, better scalability, and match the microservices architecture.