Domain-Driven Design
A Focus On Software Architecture

Eduard Ghergu, Eng., PhD.
www.professional-programmer.com

About Me

I have more than 24 years of experience in software development, being involved in various projects covering desktop, client/server, web, cloud, and mobile applications using mainly Microsoft tools & technologies, but also Apple and open-source.

I'm a Certified Scrum Master and a certified APMG Agile Project Manager. Since 2003, I'm holding a Ph.D. degree in Industrial Engineering.

You can find more about me on LinkedIn:
https://www.linkedin.com/in/eduardghergu/

Domain-Driven Design: A Deep Dive

1. A short recap on the gathered knowledge

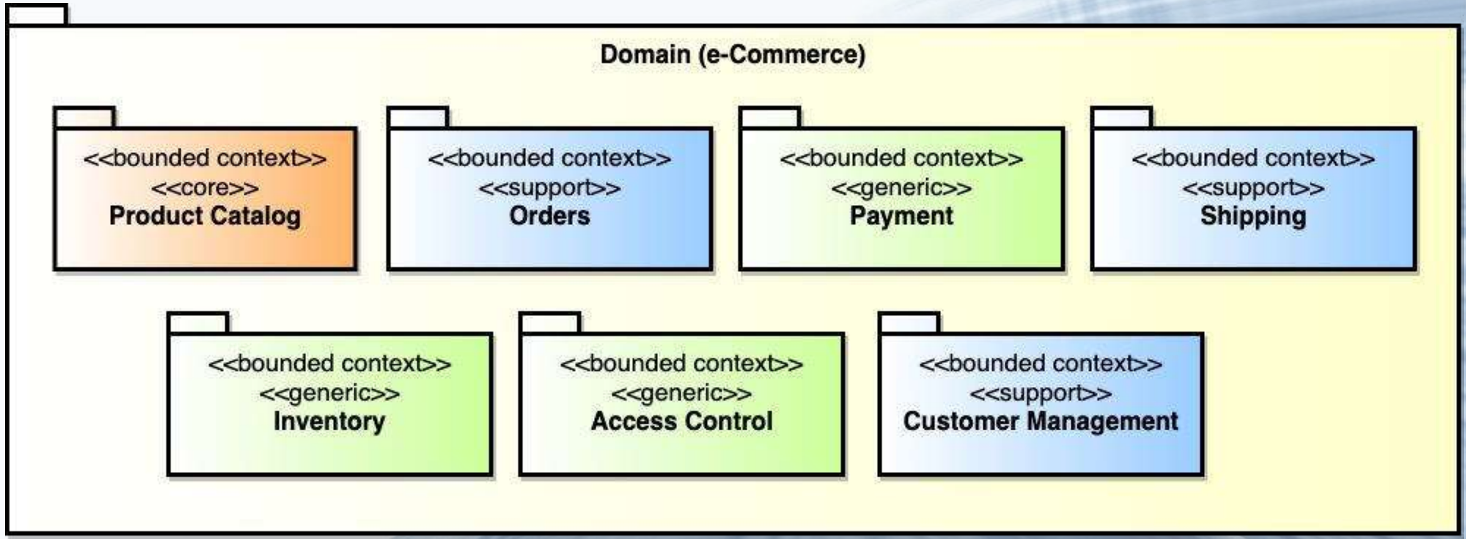
2. Continuing the work on the sample e-Commerce application

3. Conclusions. Q&A

Eduard Ghergu, Eng., PhD.
www.professional-programmer.com

The gathered knowledge

The domain modeled by e-Commerce app:


Domain (e-Commerce)
<<bounded context>> <<core>> Product Catalog
<<bounded context>> <<support>> Orders
<<bounded context>> <<generic>> Payment
<<bounded context>> <<support>> Shipping
<<bounded context>> <<generic>> Inventory
<<bounded context>> <<generic>> Access Control
<<bounded context>> <<support>> Customer Management

Eduard Ghergu, Eng., PhD.
www.professional-programmer.com

The gathered knowledge - cont.

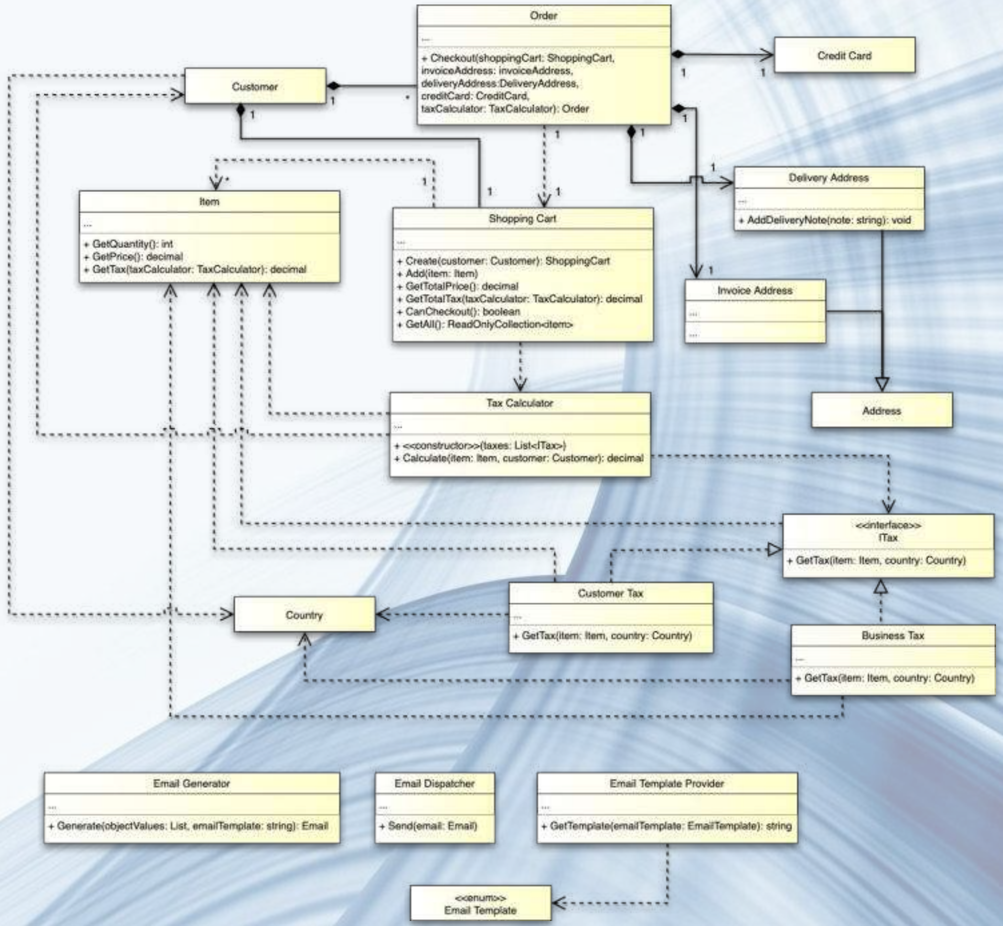
The identified Bounded Contexts are classified as Core, Support and Generic:

- Usually, there is only one Core, that represents the main part of the Domain - the ***Product Catalog***.
- Support Bounded Contexts are auxiliary ones. They will support the Core, however, they will have their own model.
- Generic Bounded Contexts are similar to support ones, but they have a strong particularity: they are pretty generic so that they could be used not only in the Domain they were created, but they could be used by others Domains, too.

Eduard Ghergu, Eng., PhD.
www.professional-programmer.com

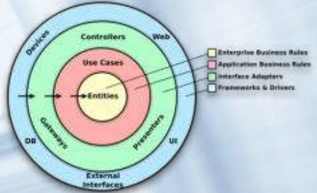
The gathered knowledge - cont.

Class diagram:

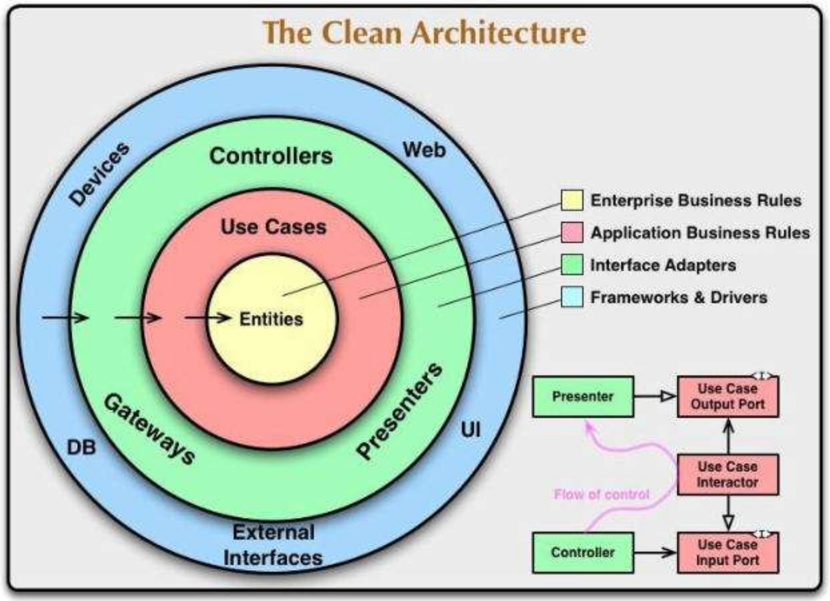

Order
...
+ Checkout(shoppingCart: ShoppingCart, invoiceAddress: invoiceAddress, deliveryAddress:DeliveryAddress, creditCard: CreditCard, taxCalculator: TaxCalculator): Order
Credit Card
Customer
Item
...
+ GetQuantity(): int
+ GetPrice(): decimal
+ GetTax(taxCalculator: TaxCalculator): decimal
Shopping Cart
...
+ Create(customer: Customer): ShoppingCart
+ Add(item: Item)
+ GetTotalPrice(): decimal
+ GetTotalTax(taxCalculator: TaxCalculator): decimal
+ CanCheckout(): boolean
+ GetAll(): ReadOnlyCollection<item>
Delivery Address
...
+ AddDeliveryNote(note: string): void
Invoice Address
...
...
Address
Tax Calculator
...
+ <<constructor>>(taxes: List<ITax>)
+ Calculate(item: Item, customer: Customer): decimal
<<interface>>
ITax
+ GetTax(item: Item, country: Country)
Country
Customer Tax
...
+ GetTax(item: Item, country: Country)
Business Tax
...
+ GetTax(item: Item, country: Country)
Email Generator
...
+ Generate(objectValues: List, emailTemplate: string): Email
Email Dispatcher
...
+ Send(email: Email)
Email Template Provider
...
+ GetTemplate(emailTemplate: EmailTemplate): string
<<enum>>
Email Template
1
*
1
1
1
1
1
1
1
1
1
1
1
*

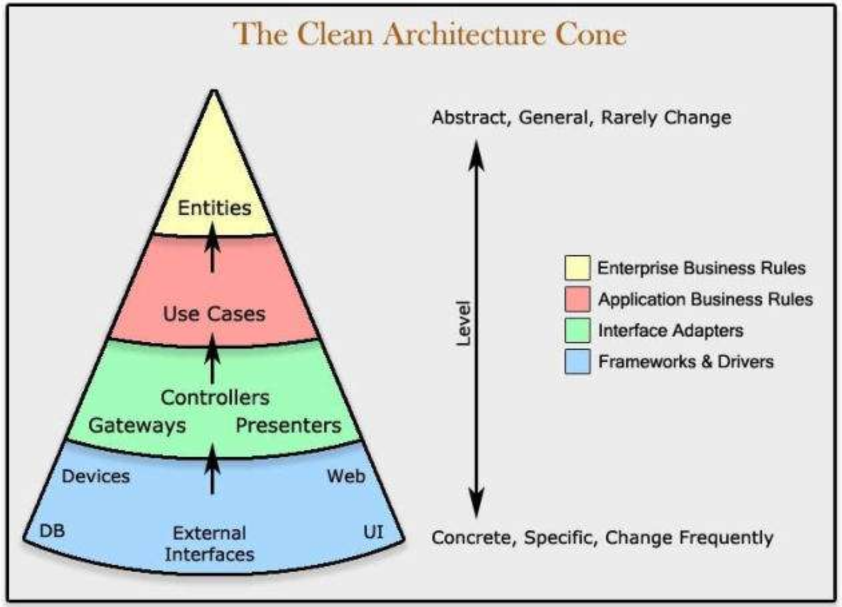
Eduard Ghergu, Eng., PhD.
www.professional-programmer.com

The gathered knowledge - cont.



Clean Architecture Building Blocks[11]



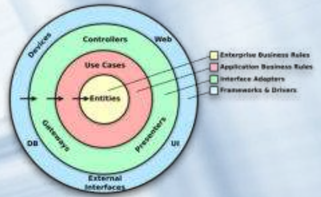


The Clean Architecture Cone
Entities
Use Cases
Controllers
Gateways
Presenters
Devices
Web
DB
External Interfaces
UI
Abstract, General, Rarely Change
Level
Concrete, Specific, Change Frequently
🟨 Enterprise Business Rules
🟥 Application Business Rules
🟩 Interface Adapters
🟦 Frameworks & Drivers

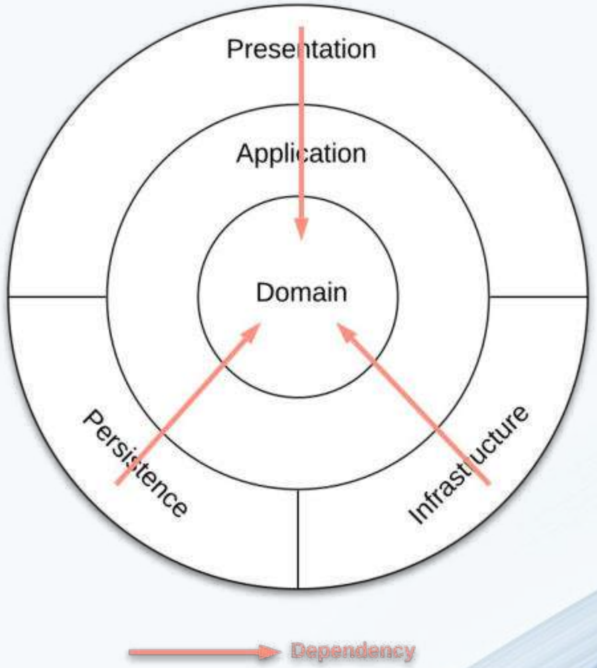
Original view [11] **Alternative view [12]**

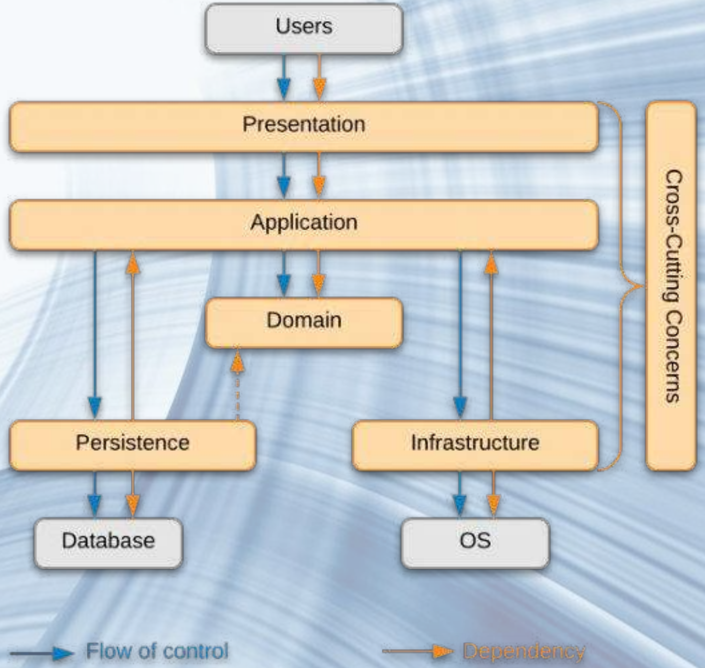
Eduard Ghergu, Eng., PhD.
www.professional-programmer.com

The gathered knowledge - cont.



Clean Architecture - Down to Earth

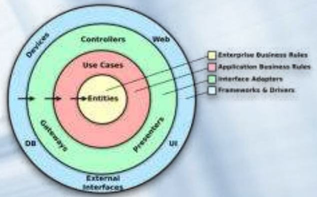




Developer view [11]

Eduard Ghergu, Eng., PhD.
www.professional-programmer.com

The gathered knowledge - cont.



Command-Query Separation (CQS)

- Software Design Pattern

Command:

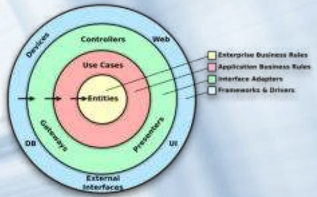
- Does something (execute a request)
- Should modify state
- Should not return a value (ideally; better, rise an event)

Query:

- Answers a question (respond to a request)
- Should not modify state
- Always returns a value

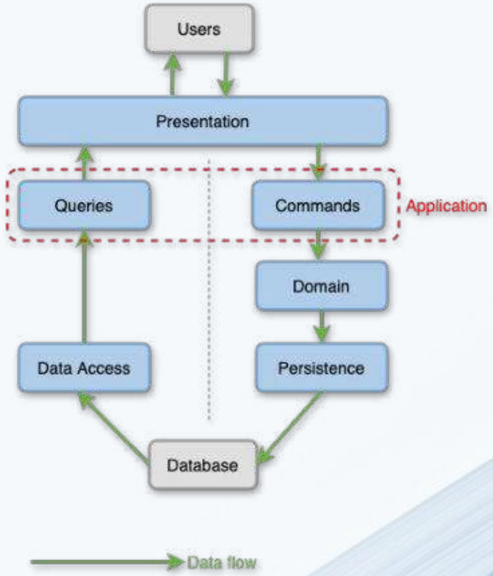
Eduard Ghergu, Eng., PhD.
www.professional-programmer.com

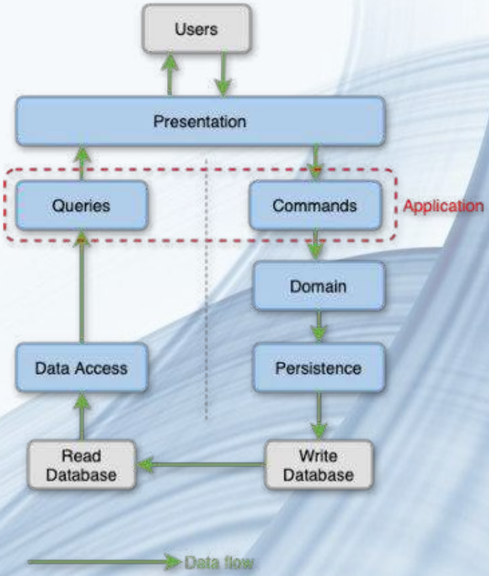
The gathered knowledge - cont.

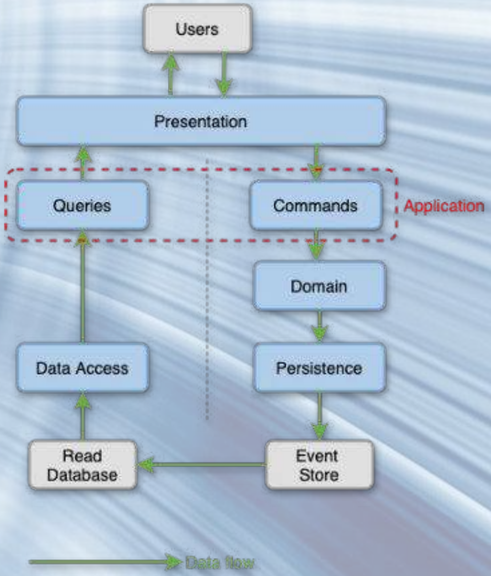


Command-Query Responsibility Segregation (CQRS)

- **Architecture Design Pattern**







Eduard Ghergu, Eng., PhD.
www.professional-programmer.com

Sample app - Code (draft)

https://github.com/AbstractSoft/eCommerce

Eduard Ghergu, Eng., PhD.
www.professional-programmer.com

Q & A

For any inquiries or requests:
eduard.ghergu@professional-programmer.com

Eduard Ghergu, Eng., PhD.
www.professional-programmer.com

Thank you!

Eduard Ghergu, Eng., PhD.
www.professional-programmer.com

References

1. https://www.infoq.com/minibooks/domain-driven-design-quickly/
2. https://vladikk.com/2016/04/05/tackling-complexity-ddd/
3. https://en.wikipedia.org/wiki/Domain-driven_design
4. https://www.jamesmichaelhickey.com/clean-architecture/
5. https://blog.knoldus.com/is-shifting-to-domain-driven-design-worth-your-efforts/
6. https://thedomaindrivendesign.io/
7. https://www.culttt.com/2014/11/12/domain-model-domain-driven-design/
8. https://martinfowler.com/bliki/BoundedContext.html
9. https://github.com/zkavtaskin/Domain-Driven-Design-Example

Eduard Ghergu, Eng., PhD.
www.professional-programmer.com

References - cont.

10. https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html

11. https://www.freecodecamp.org/news/a-quick-introduction-to-clean-architecture-990c014448d2/

12. http://www.fabriciosuarte.com/2016/02/domain-driven-design-hands-on-example.html

Eduard Ghergu, Eng., PhD.
www.professional-programmer.com