Capability Based Approach to Microservices Design

Prof. Dr. Galal Hassan Galal-Edeen
Hassan Ahmed Hassan, PhD
Agenda

- Problem Definition
- Monolithic Architecture and Challenges
- Characteristics of Microservices Architecture
- Domain Driven Design Approach to Microservices Composition
- Business Architecture Approach to Microservices Composition
- Case Study and Approach
Problem Definition

• Microservice architecture is being utilised to create enterprise software systems that are flexible, agile, robust, and scalable. It's a method of building a single software system as a collection of services, each of which runs in its own process and communicates using lightweight protocols.

• However, in order for the microservices approach to properly solve this issue while avoiding a worsening of the situation, it must employ an effective method for componentizing the software system into stable services composition.

• The issue is that determining exactly where the component borders should be is difficult.

• This note addresses this issue by defining a method for designing robust microservices composition utilising the business capabilities.
Monolithic Architecture

• *The monolithic architecture is an architectural style that structures the application as a single executable component*
Composition Design Principles

01  Modularity
- Reduced complexity
- Decentralised system design
- Improved stability, adaptability, and maintainability

02  Information Hiding
- Hide as many details as possible behind a module boundary
- Communicate through interfaces

03  Cohesion
- Relatedness of module components
- Core function and is indivisible
- Changes made in as few places as possible

04  Coupling
- Degree of interdependence between pairs of modules
- Low coupling
Microservice Architecture

• The microservice architecture is an architectural style that structures an application as set of deployable/executable unites called services
  • A service is:
    • Highly maintainable and testable
    • Minimal lead time (time from commit to deploy)
    • Loosely coupled
    • Highly coherent
    • Fault Isolation
    • Independently deployable
    • Mixed technology stacks
    • Implements a business capability
    • Owned/developed/tested/deployed by small team
Domain Driven Design

Subdomain
• Subdomains resemble sets of interrelated, coherent use cases. Such sets of use cases usually involve the same actor, the business entities, and they all manipulate a closely related set of data.
• Core, Generic and Support

Ubiquitous Language
• Language of the business domain.
• The ubiquitous language must be precise and consistent. It should eliminate the need for assumptions and should make the business domain’s logic explicit.

Model
• A model is a simplified representation of a thing or phenomenon that intentionally emphasizes certain aspects while ignoring others. Abstraction with a specific use in mind.

Bounded Context
• Bounded contexts are the consistency boundaries of ubiquitous languages.
• A language’s terminology, principles, and business rules are only consistent inside its bounded context.
The problem with this approach is how to identify the business subdomains and bounded context. DDD describes a tactical technique for defining these notions known as "Event Storming." However, this method is unconnected to a larger company framework of strategy, business model, and operating model. Consequently, the resulting microservice decomposition will be vulnerable to strategy misalignment, limited business reusability, and a reduced ability to adjust to strategic changes.
Business Capability as Context for Microservices Composition

- Business capability has various properties that make it a good context for the microservices decomposition:
  - In comparison to other aspects of the business, it is stable. A one-hundred-year-old insurance firm, for example, would have had capabilities comparable to those available today: Customer Management, Policy Management, and Claim Management. While these capabilities, would not have had automation a century ago, they still existed.
Business Capability as Context for Microservices Composition

- Business capability has various properties that make it a good context for the microservices decomposition:
  - It is technology and organization structure agnostic. This means the underlying technology and organization structure can change for the same capability. This creates a stable border for the microservices, allowing them to change their implementation without affecting the boundary itself.
Business Capability as Context for Microservices Composition

• Business capability has various properties that make it a good context for the microservices decomposition:
  • Business capability is a high-level concept and can be linked unambiguously to the business strategy.
  • This provides a strategic context for the microservices composition. The development of business capabilities, as well as their deployment to produce value, can be impacted by business strategy. This makes the microservices composition strategically agile. It can be linked to the strategy through the business capability. So, strategy can either change the implementation of the microservices or change the its wiring, without changing the microservices' boundaries. This makes the microservice composition both stable and agile.
Business Capability as Context for Microservices Composition

- Business capability has various properties that make it a good context for the microservices decomposition:
  - Business capabilities are applied in a variety of business scenarios and value generation contexts, making them a reusable notion across the firm. This allows microservices to be utilized across the organization and results in a simple microservices decomposition.
Business Capability as Context for Microservices Composition

- Business capability has various properties that make it a good context for the microservices decomposition:
  - Microservices architecture style takes on a strategic dimension when they're aligned with business capabilities. Through the line of sight from business strategy to business capabilities to microservices that enable those capabilities, the impact of strategy on software systems becomes obvious and clear. From another perspective, the microservices architecture style's agility and adaptability make the strategy easier to implement.
Case Study

• A business scenario in the banking industry was used to test this microservices decomposition approach. Banks offer credit to many types of consumers, including retail and business customers. The bank does numerous operations to onboard the customer, screen the consumer, structures a deal, undertake risk due diligence, give limits, authorize the deal, collect paperwork, sign contracts, activate lines, and utilize facilities in order to establish a credit arrangement with the customer.
Establish and Activate Credit Deal

- Customer: Defined, Established, Screened and Authenticated
- Credit Agreement: Requested
- Credit Agreement: Established, Initially Structured, Initially Priced
- Credit Agreement: Structured, Priced and Approved
- Credit Agreement: Risk Assessed, Risk Mitigants are Identified
- Credit Agreement: Associated Terms, Conditions, Covenants and Collaterals are Defined
- Credit Agreement: Documentation and Contracts Defined, Created, Signed, Collected and Stored

Stakeholder: Business Customer
- Stakeholder: SME
- Stakeholder: Corporate Customer

Credit Agreement: Activated on Core Systems

Customer or other requester has new or updated credit deal (agreement).
Customer Management Capability
Customer Management to Microservices Decomposition
Agreement Management to Microservices Decomposition

Agreement Application Management
Agreement Definition
Agreement Identification
Agreement Establishment

Agreement Application Microservice

Agreement Profile Management
Agreement History Management
Agreement Analytics and Related Prediction
Agreement Preference Management
Agreement Eligibility Validation

Agreement 360 and Analytics Domain

Agreement Eligibility and Compliance

Agreement Product Structure Configuration
Agreement Term Management
Agreement Portfolio Management

Agreement Products Structure Microservice
Agreement Terms, Conditions and Covenants
Agreement Portfolio Management Microservice

Agreement Compliance Determination

Agreement Type Management
Agreement State Management

Agreement Pricing Microservice

Agreement Collateral Determination

Agreement Product Limit Determination

Agreement Lifecycle
Agreement Activation
Agreement Termination
Agreement Renewal

Agreement Lifecycle Management Microservice

Agreement Content and Documents Management

Agreement Limit Management Microservice
Agreement Content Management
Next Actions

• Build the enterprise capability map to at least the third level.
• Build the value stream map and start with the most important customer facing one. Map the selected value streams to the enabling capabilities.
• Build the information map.
• Design the microservices boundaries using the capability map and the information map.
• Design the microservices interaction using the valuestream/capability cross mapping.
Conclusion

• The enterprise architecture is introduced in as a framework that provides the appropriate components for abstracting business domains and bounded contexts with characteristics that can be used to define microservices boundaries and good microservices decomposition.

• Business capabilities have the characteristics of being stable, technology and organization structure agnostic, a high-level concept and can be linked unambiguously to the business strategy, and able to transform value generated in wide business scenarios.
Conclusion

• A good microservices decomposition follows the principles of good structural decomposition including modularity, information hiding, high cohesion and low coupling. So, we need a decomposition approach that realizes those principles.

• Domain driven design has been introduced as an approach to microservices decomposition. It’s notions of business domains and bounded contexts within those domains are presented as reasonable guide to create the microservices boundaries. However, this approach lacks the business context of a larger framework of strategy, business model, and operating model.
THANK YOU