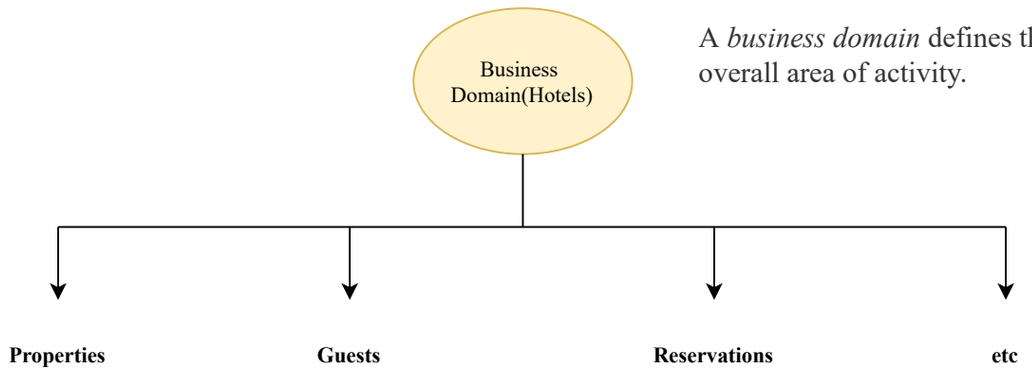
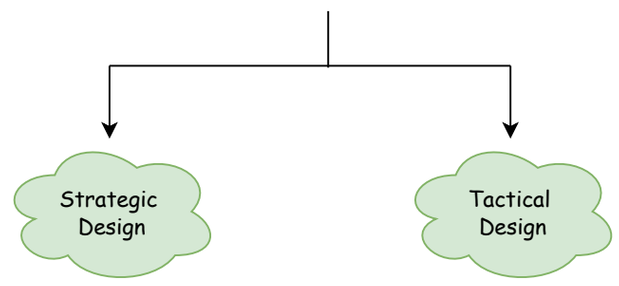
Microservices Architecture



A *business domain* defines the company's overall area of activity.

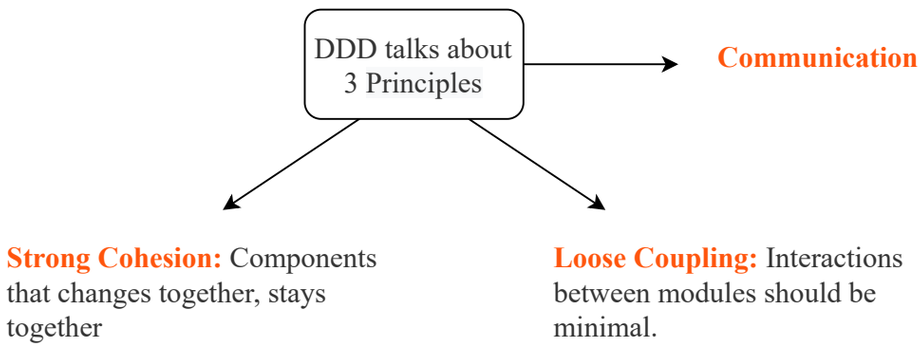
A subdomain is a fine-grained area of business activity. All the subdomains together add up to the company's business domain

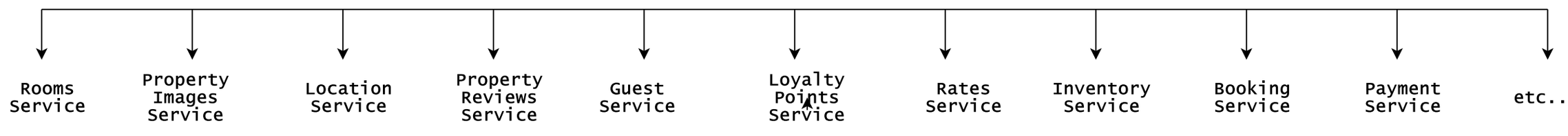
DDD is a methodology that covers the whole software development lifecycle, from gathering requirements to low-level design. Its core premise is that to build better software, we have to align its design with the business's domain, needs, and strategy. That's where the name comes from: (business) domain-driven (software) design.



Strategic design provides a framework for decomposing a business domain(System) into modular components and mapping the interactions between them.

Tactical design is to architect the solution so that it fits the problem domain as closely as possible.





Why MicroServices ??

SOA (Service-Oriented Architecture): It is a style of software design where services are provided to other components by application components through a communication protocol over a network.

Mainly concerned about Modular programming and Protocol to communicate over network(SOAP or REST).

Which leads to problems:

1. Monolith Systems [All functionality in a system must be deployed together]
2. Vertical Scaling [It is not possible to scale a particular component]
3. Deployment [A small code change requires, entire application to be deployed]

MicroService architecture is a part of SOA but design enhancements to SOA.

Primary Goals in MicroService Architecture:

1. Independent Deployability
2. Modelled around a business domain
3. Owning their own state

Basically, ***Microservices are independently releasable services that are modelled around a business domain.***

Microservices Advantages:

1. Technology Heterogeneity. [Pick right tool for each service.]
2. Robustness. [One component failure, will not impact other components]
3. Scaling. [Horizontal scaling per service.]
4. Ease of deployment.
5. Organizational Alignment. [Smaller teams and focused on one specific component]

Microservices Pain Points:

1. Data Consistency
2. Monitor & Troubleshoot
3. Management of many hundreds of services
4. Impact on Data analytics platform [Aggregate data from many databases]

Challenges:

1. Decompose monolithic system to subdomains [DDD Design]
2. Understand the relationship between Microservices and DDD
3. How to model microservices
4. Microservices communication styles [Sync vs Async vs Request-Response vs Queue vs Topic]
5. Implementation technologies and best practices [Restful over HTTP, GraphQL, gRPC, RPC]
6. Get consensus across domains on principles/standards on how to build microservices across organization

Microservices Implementation Technologies & Best Practices

Available technologies to implement microservices:

- Restful over HTTP
- RPC
- gRPC
- GraphQL