

# Scalable Architectures

© 2008 Huascar A. Sanchez. All rights reserved.

By

M.E. Fayad & H A. Sanchez

# Agenda

- Scalability – what's it?
- Types of Scalability
  - Scaling Up & Down
  - Scaling In & Out
- Scalability is a major concern
  - Unfortunately...
  - In short...
- Conventional Scalability.
- Scalability with stability in mind.
  - Software stability.
  - Scalable architectures.

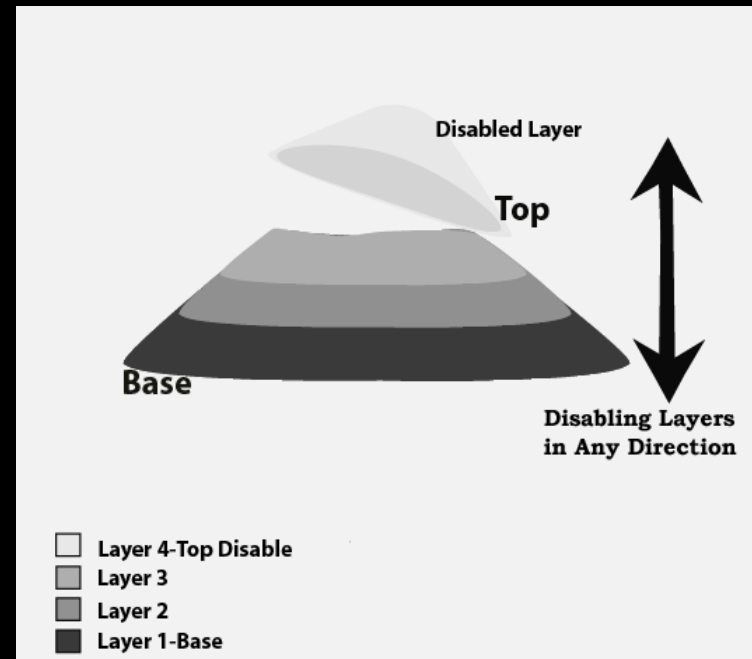
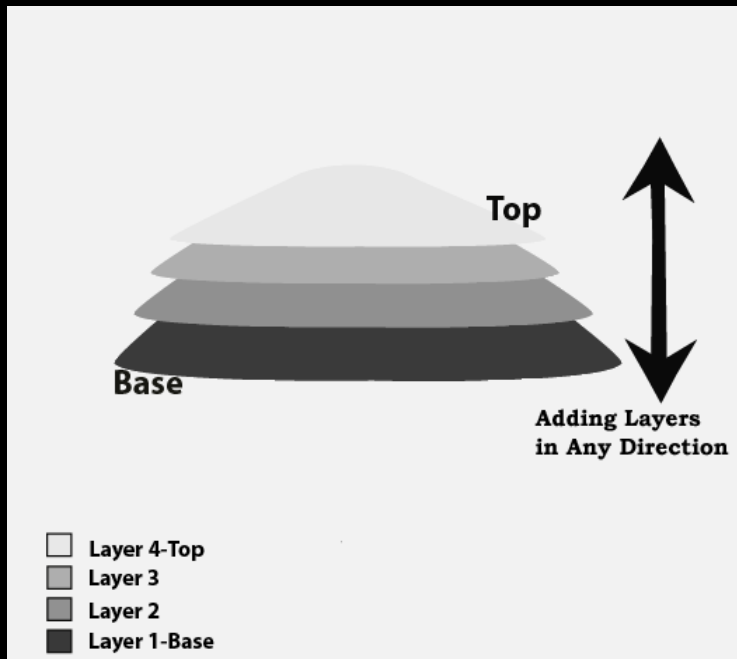
# Scalability - What's it?

- *Scalability* indicates the ability of architectures to easily adapt to evolving requirements without unnecessary effort.
- Scalability can be further divided into two kinds: *Vertical* and *Horizontal* scalability.

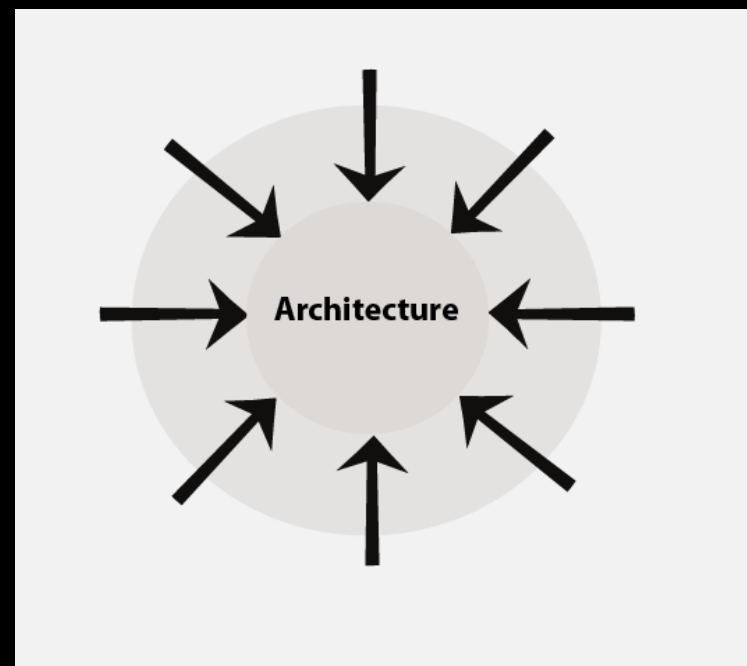
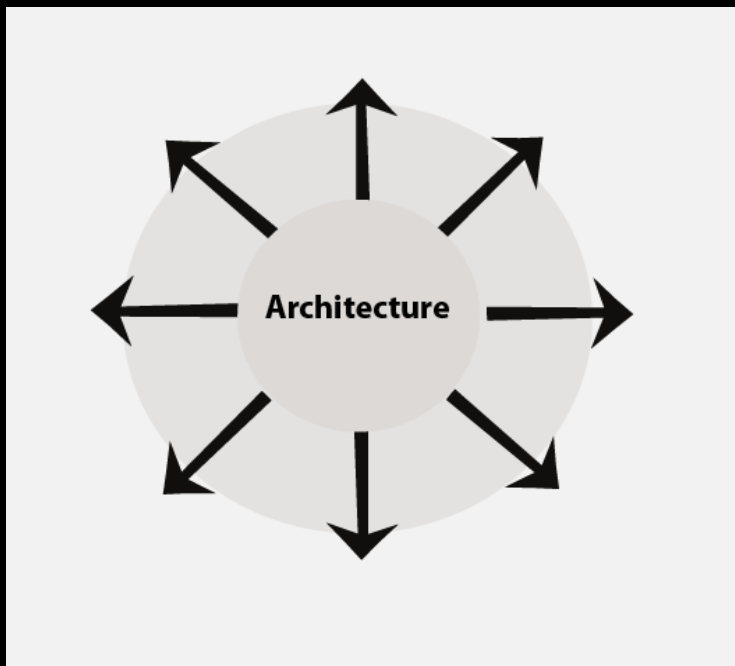
# Types of Scalability

- *Vertical Scalability* is concerned with the ability of architectures to scale **Up** and **Down** in order to adapt to evolving requirements.
- *Horizontal Scalability*, on the other hand, emphasizes the ability of architectures to scale **In** and **Out** in an efficient manner.

# Scaling Up & Down



# Scaling In & Out



# Scalability is a major Concern

- In most cases changes that cause software architectures to fail are:
  - (1) Natural evolution of an architecture business processes.
  - (2) Modification of underlying requirements to meet evolving needs.
- When Business experience major increments in services demands, the main concern is their architectures' ability to scale and accommodate these loads. Hence:
  - (1) Their architectures need to efficiently scale in any direction and adapt to handle these load variations.

# Unfortunately...

- Scalability in software architectures are emphasized merely on the achievement of *Vertical Scalability*; especially one of its directions: *Up*.
- They solely rely on three elements: increasing speed and capacity, improving efficiency, and shifting or reducing the loads (i.e., load balancing mechanism).
- Their validity usually relies on special hardware.

# In short...

- Achieving full scalability in software architectures is not straightforward, and especially with current approaches.

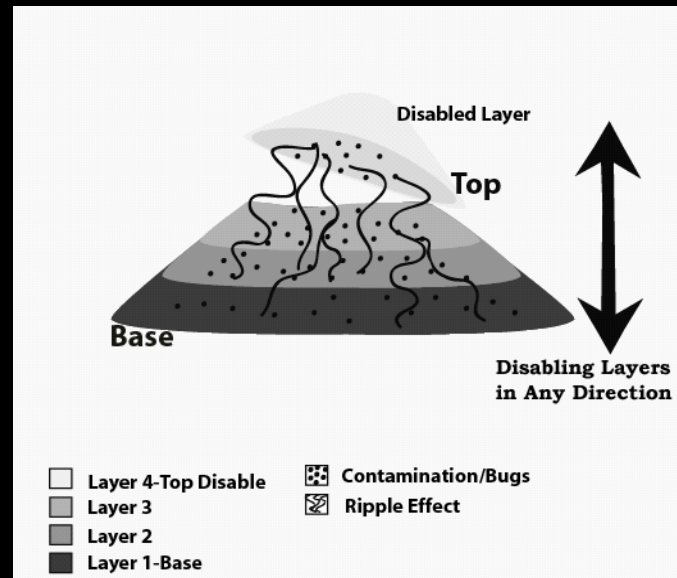
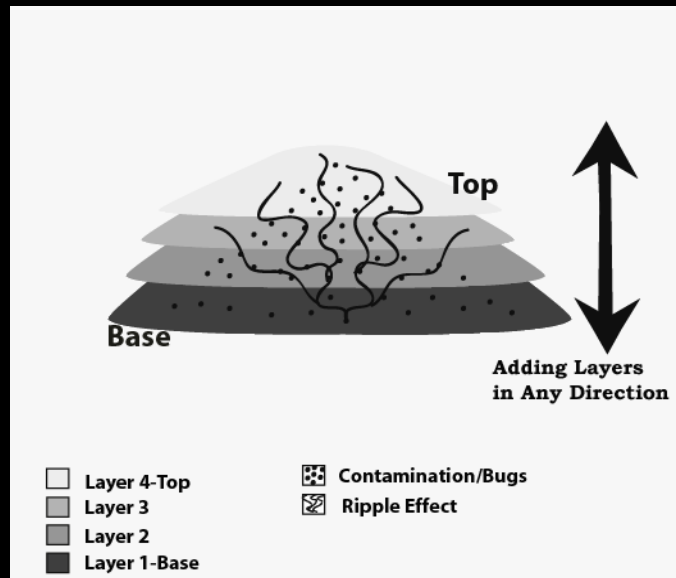
# Conventional Scalability

- Common problems

Scalability	Cause		Effect	
	Guidelines	Processes	Implementation	Architecture
	Availability	Usage	Impact & Complexity	Occurrence
Up & Down	1,2,3,4 - None	1,2- High	1,2,3 – High Impact 4- Med. Effort/High Impact 5- High Effort & Impact	1,2,3- High
In & Out	1,2,5- None	1,2- High	1,2,3- High Impact	1,2,3 – High
Problem Definition	1.No Methodology 2.Unclear Direction 3.No Layer Identification 4.No Connection Points 5.Layer Boundary Shortage	1.Non-Systematic Process 2.Ad-Hoc	1.Impossible test scenarios. 2. Low Cohesion/Unstable Structure 3.No Cost/Time Effective 4. Scaling Up 5. Scaling Down	1.Architecture Collapse 2.Embedded Contamination 3.Ripple Effects

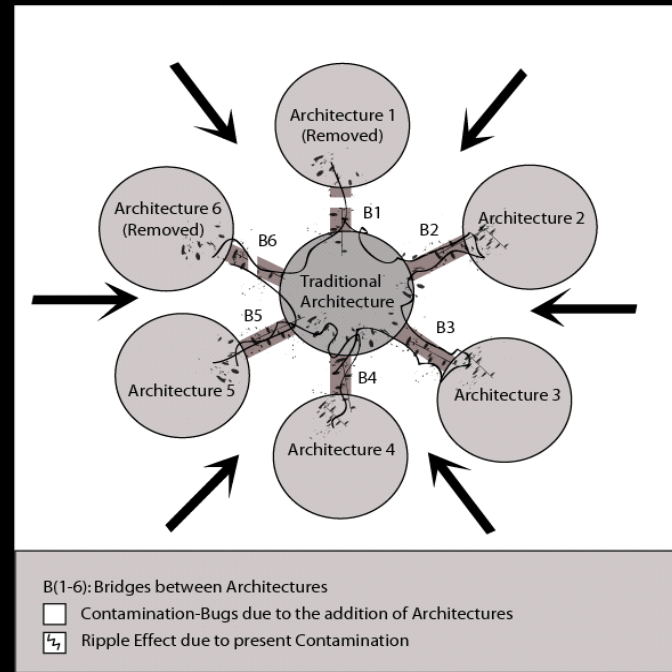
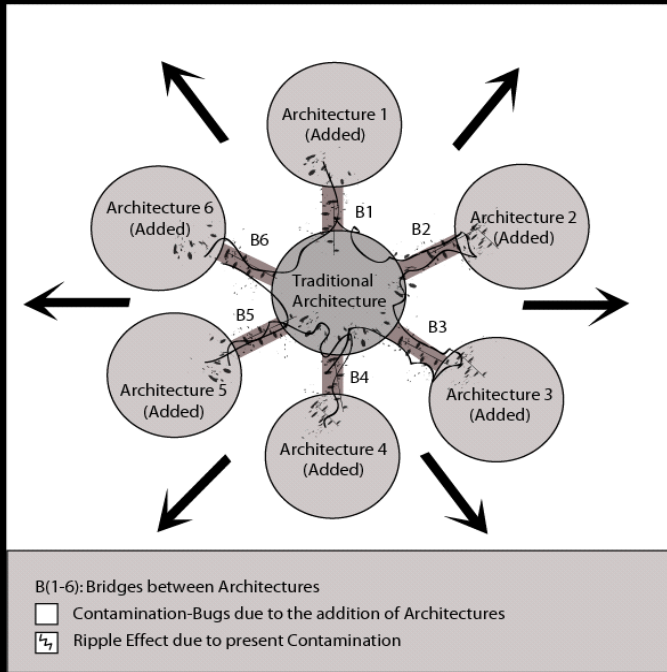
# Conventional Scalability

- A different view



# Conventional Scalability

- A different view

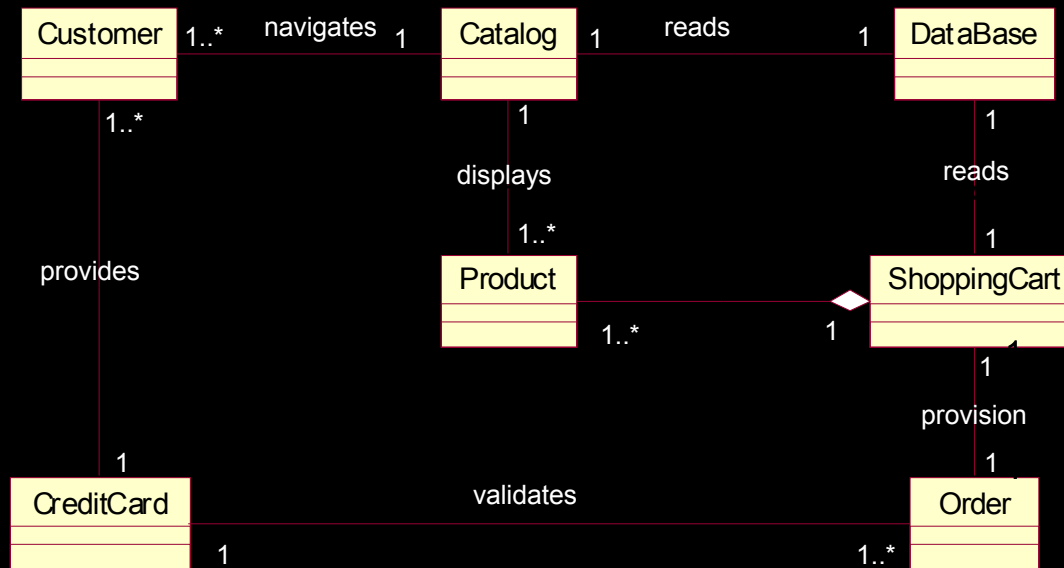


# Conventional Scalability

- Case study: E-commerce application
- The architecture contains the following objects: Customer, Catalog, Database, ShoppingCart, Product, Order, CreditCard.

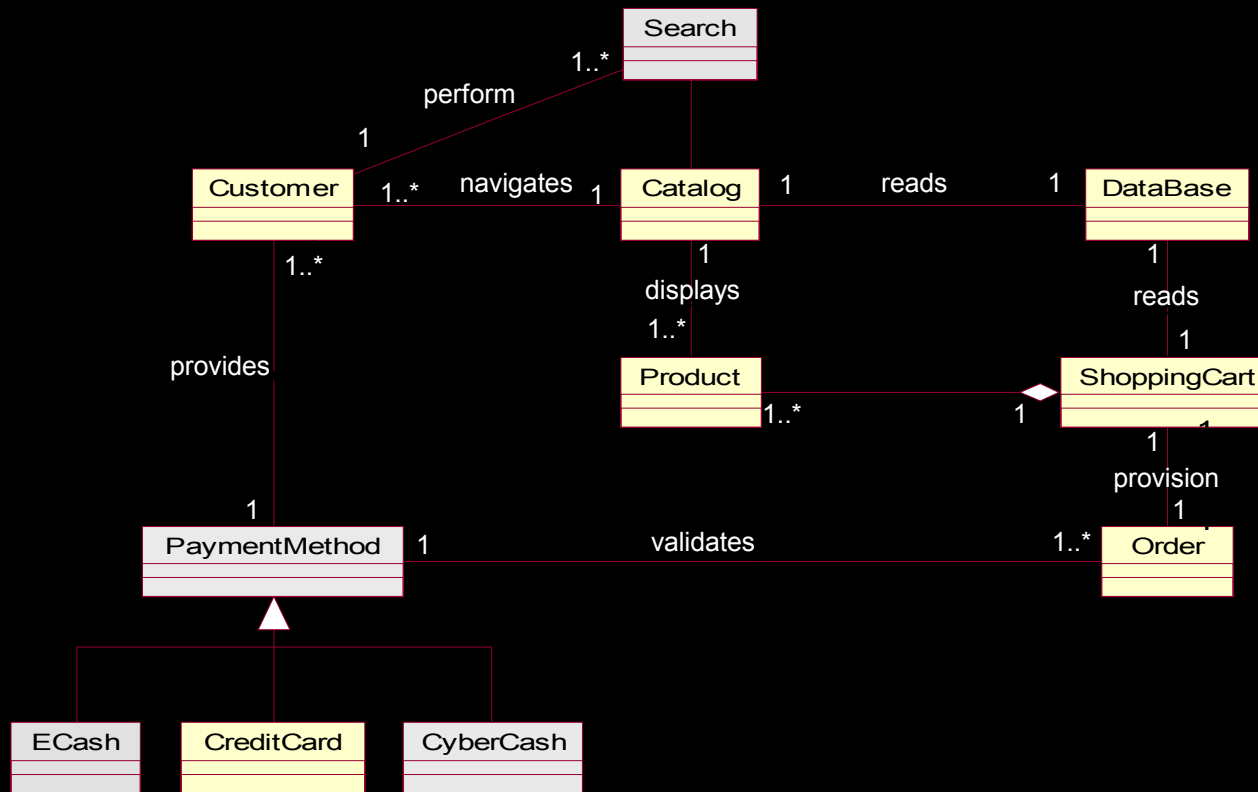
# Conventional Scalability

- E-commerce application class diagram



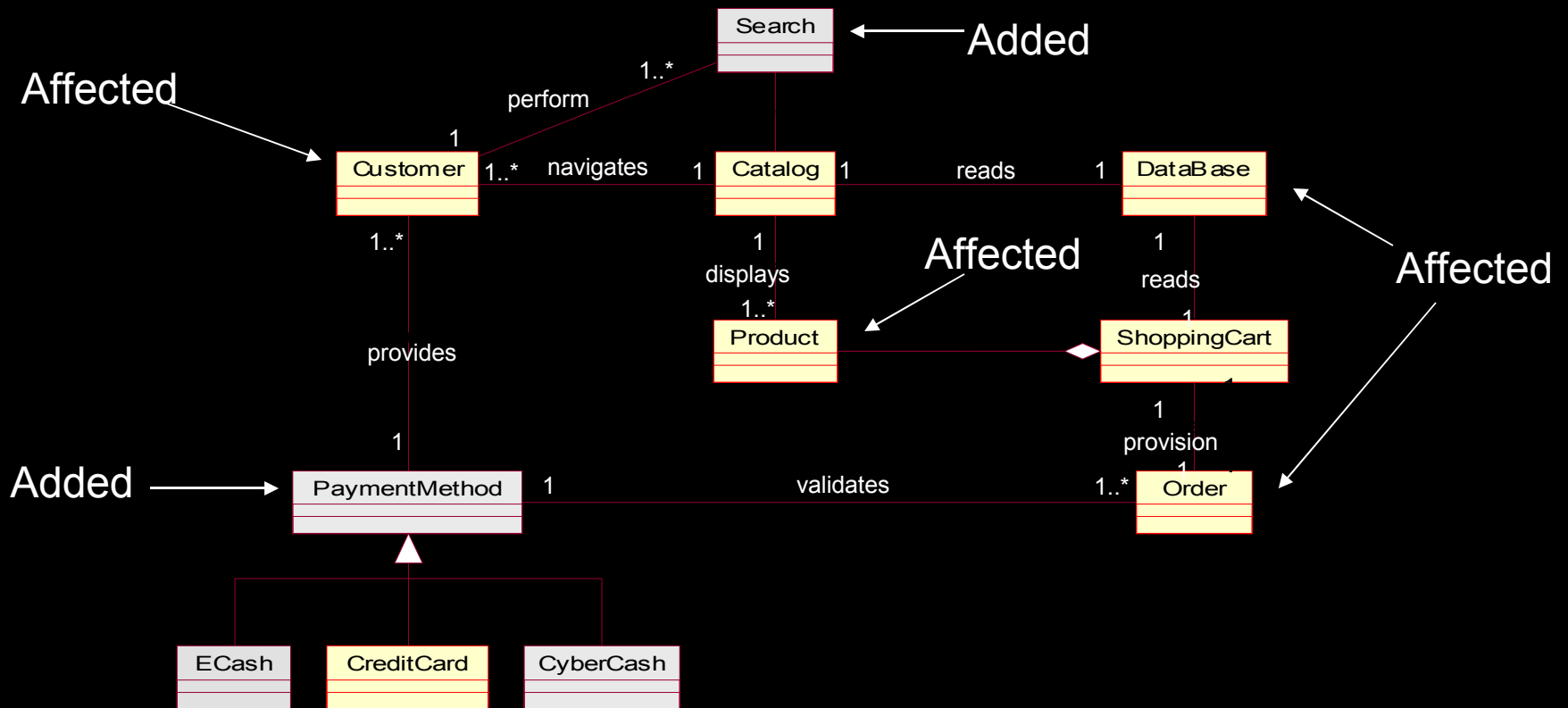
# Conventional Scalability

- E-commerce application class diagram



# Conventional Scalability

- Same app after changes



# Scalability with Stability in mind

- Software Stability Model (SSM) as the means for developing scalable software architectures.
- These architectures are flexible enough to scale the scope of their elements, methods, etc. when handling evolving requirements

# Scalability with Stability in mind

- They provide the means for gluing two or more architectures to perform one or more common tasks (Horizontal Scalability), or
- Adding new functionality to the architecture's structure to address more users needs (Vertical Scalability).

# Scalability with Stability in mind

- In short, these architectures are
- 
- Capable of evolving through time without any concern of a potential collapse.
- Referred to as *Scalable architectures*.

# Scalability with Stability in mind

- Scalable architectures *are well-designed architectures, which structures remain constant, and are able to evolve proportionally with the introduction of new requirements.*

# Scalability with Stability in mind

- Software stability guarantees the identification of changeable components in the software architecture.
- It separates them from the stable elements.
- Stable: less like to change over time.

# Software Stability?

- Software stability classifies a software architecture into three layers: EBTs, BOs, and IOs.
- EBTs are conceptual classes, which structure remain internally and externally stable.
- BOs are classes that are internally stable but externally adaptable.
- IOs are context specific classes, which structure is volatile (internally and externally unstable).

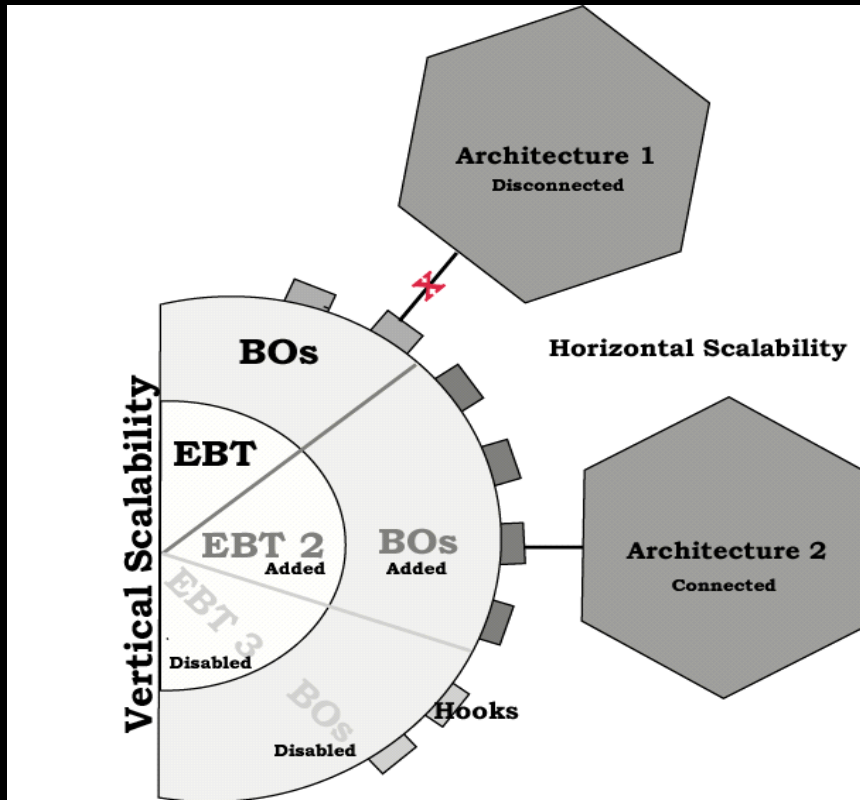
# Software Stability?

- The synergy between EBTs and BOs form the basis for Scalable Architecture.
- IOs are attached to the scalable architecture by means of certain extension points of the BOs, referred to as “Hooks.”

# Software Stability?

- These extension points can be added to or removed from a BO according to the problem needs and requirements.
- BOs serve as the linking points for architectures to establish a connection with external architectures (Horizontal Scalability).

# Scalable architectures



Properties granted to Software Architecture:

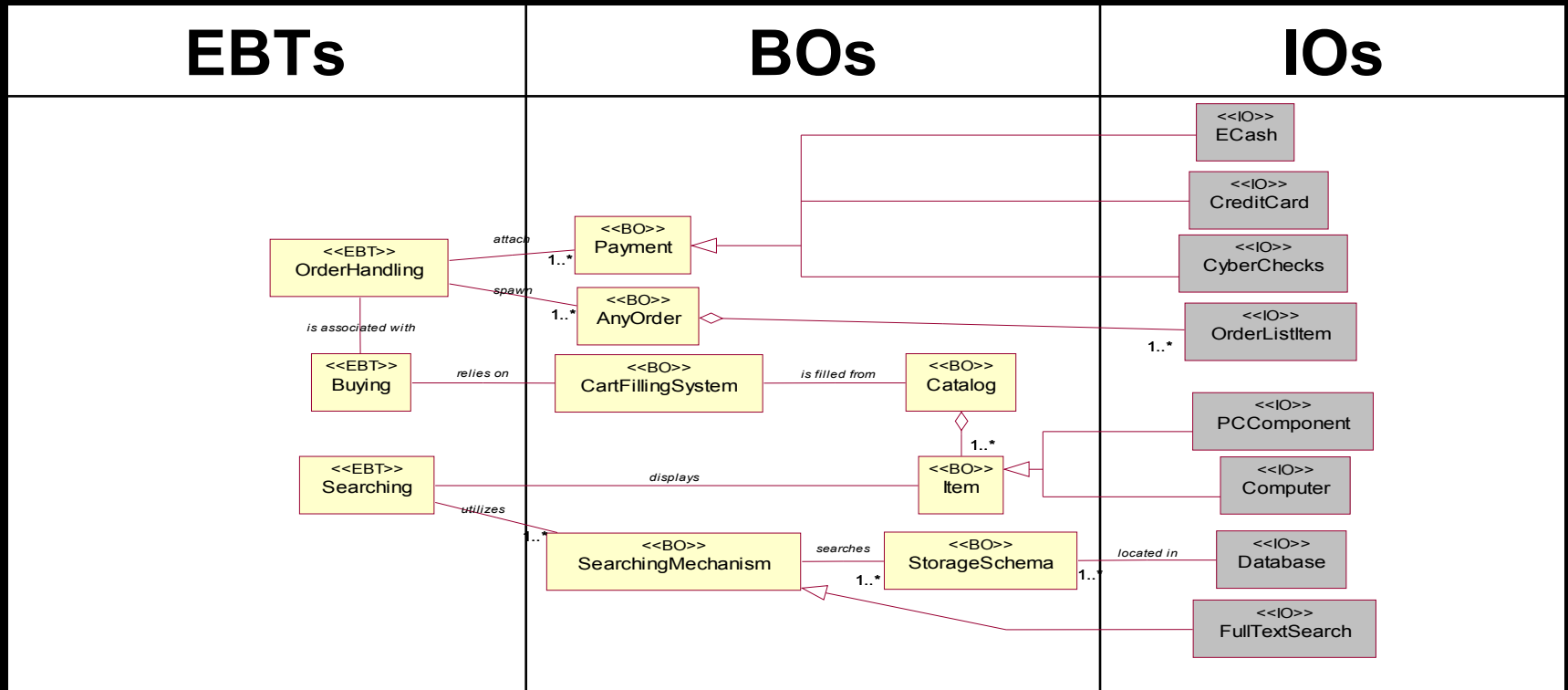
1. Stability
2. Scalability
3. Adaptability
4. Customizability
5. Traceability
6. Separation of Concerns
7. and more.....

# Scalable architectures

- Case study: E-commerce Application
- Identification Process:
  - First, we need to identify the purpose, and goals of this system (EBTs).
  - Then, we identify the core abstractions of the system' processes (BOs).
  - Finally, we need to identify the physical entities of our model (IOs).

# Scalable architectures

- E-commerce Application: Stable Model

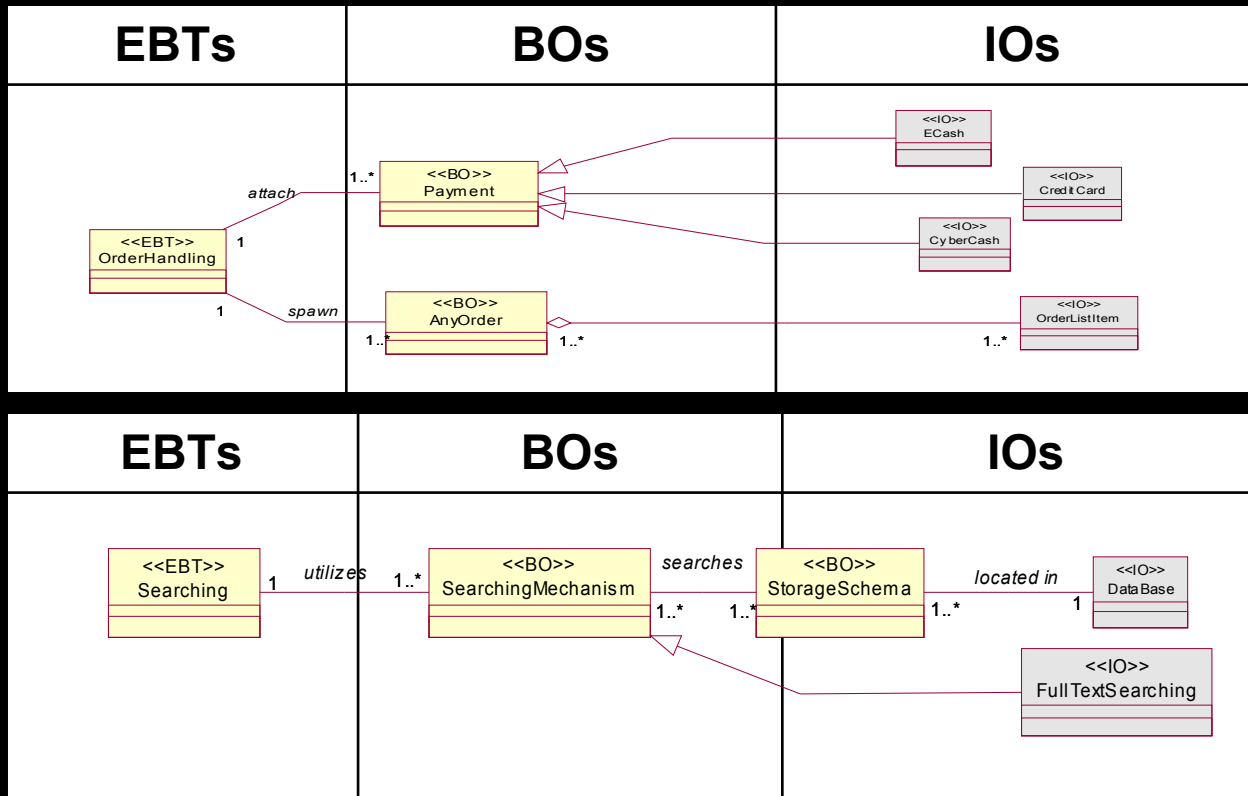


# Scalable architectures

- Any single representation of EBTs along with their associated BOs and IOs, represent an independent functionality of the group of functionalities in the entire system's architecture.
- Each functionality, by its definition, is self-controlled. Nevertheless, it collaborates with other functionalities in order to accomplish one or more tasks in the architecture.

# Scalable architectures

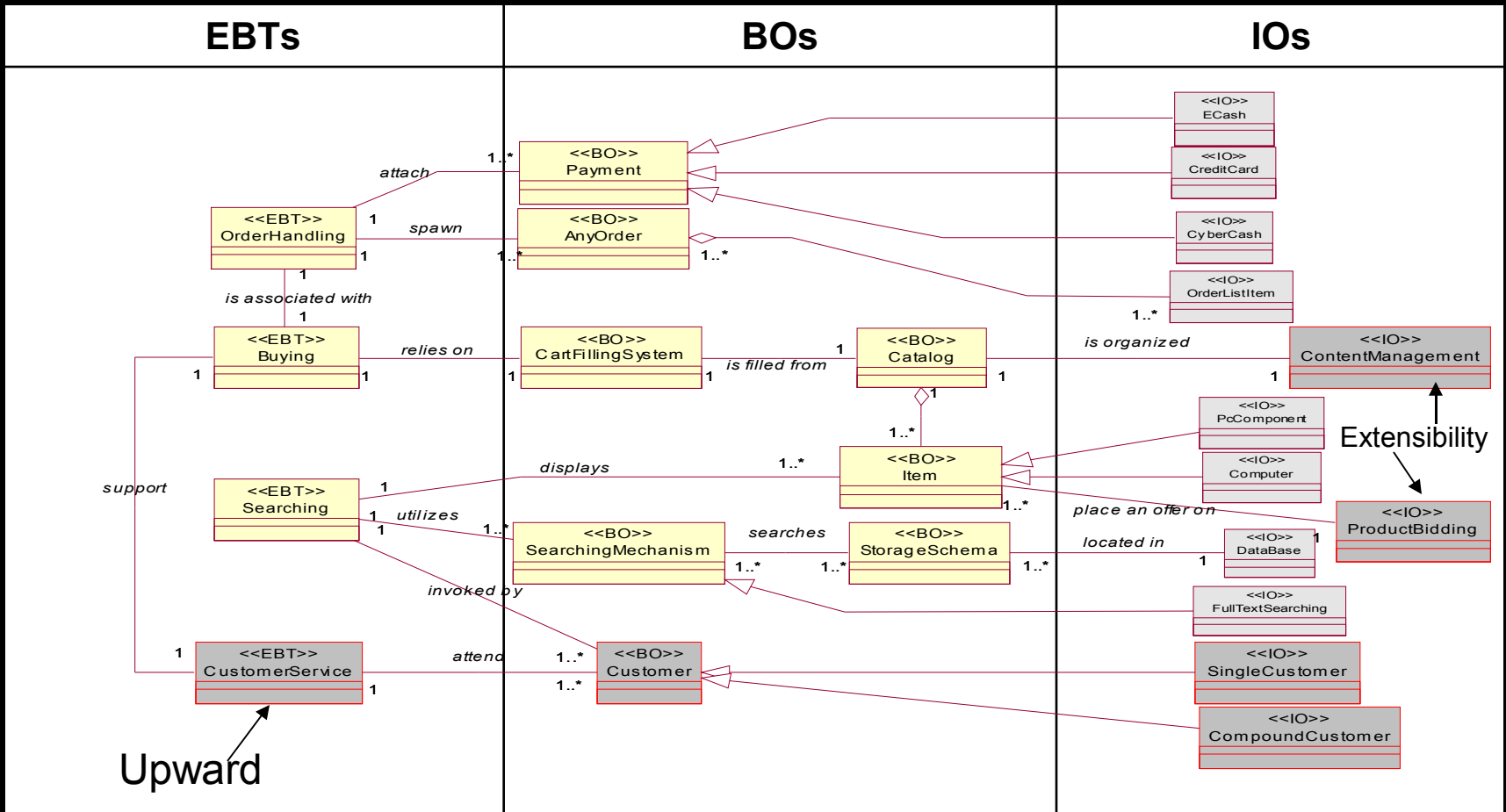
- Independent functionality



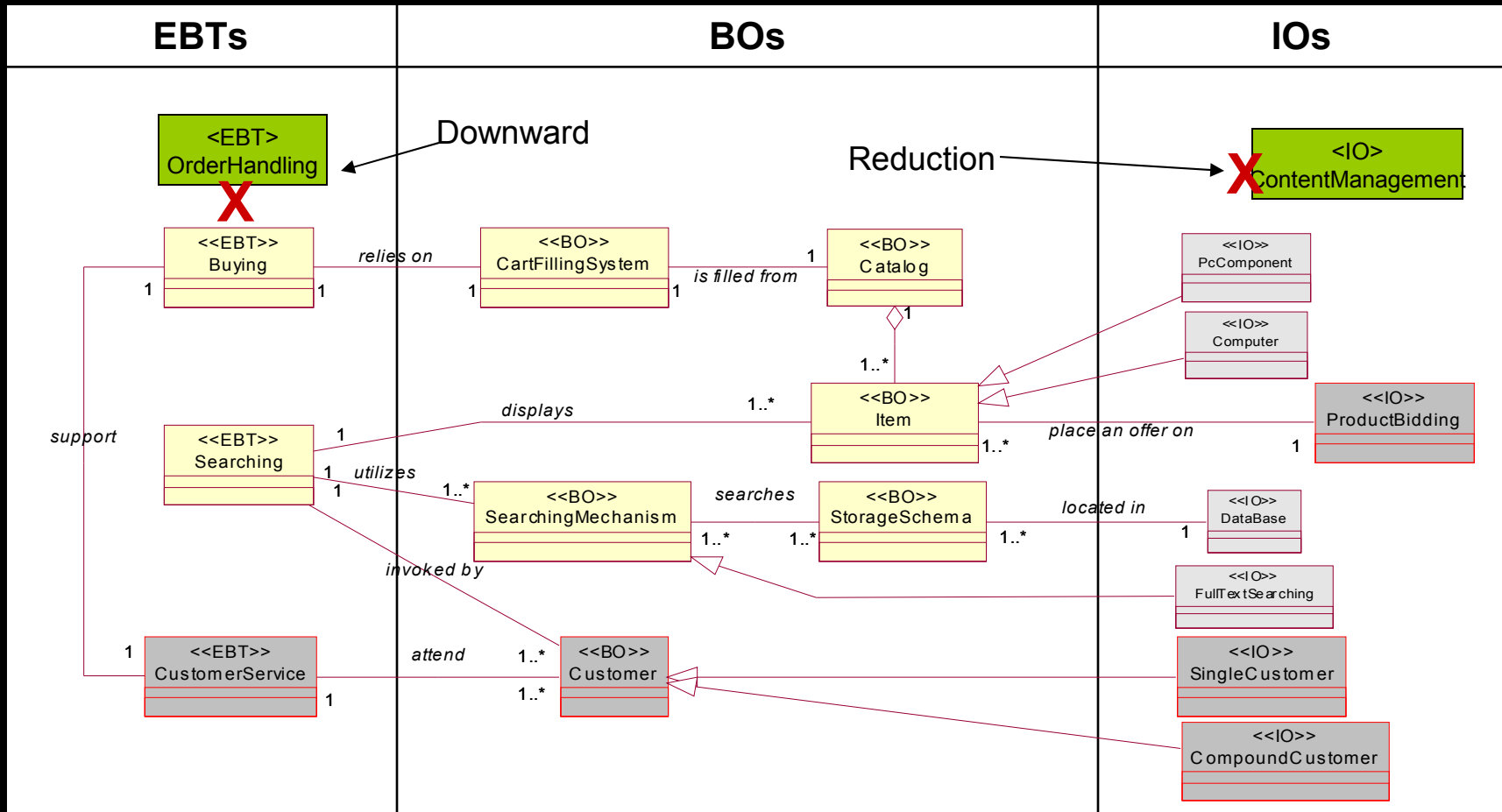
# Scalable architectures

- Vertically & horizontally scaling.
- Adding Customer Service, Content Management, Bidding Capacity
- Disabling Order Handling, and Content Management Application

# Scalable architectures



# Scalable architectures



# Conclusions

- We have shown, through a simple case study, that developing architectures using the traditional approach does not allow the desired scalability to accommodate future requirements.
- On the other hand, software stability approach partitions the entire functionality into several independent layers, and hence, we are able to incorporate scalability feature into the architectures.
- Consequently, architectures can scale up or down, and scale out or in to accommodate evolving requirements, without the danger of being collapsed.
- Such flexibility will improve the quality of the developed architectures while concurrently diminishing the cost and condensing the time of the development process.

Questions?