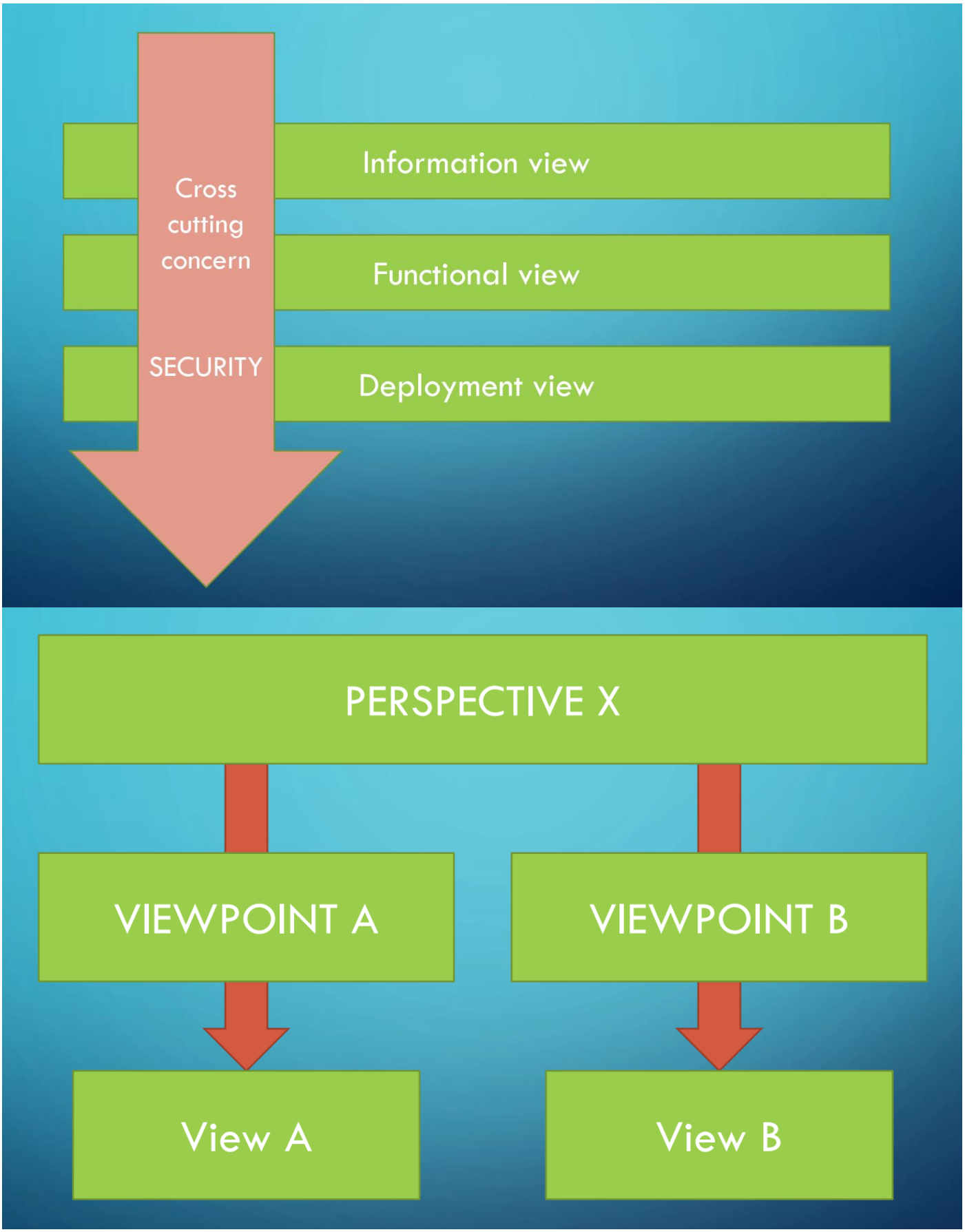
ARCHITECTURAL PERSPECTIVES

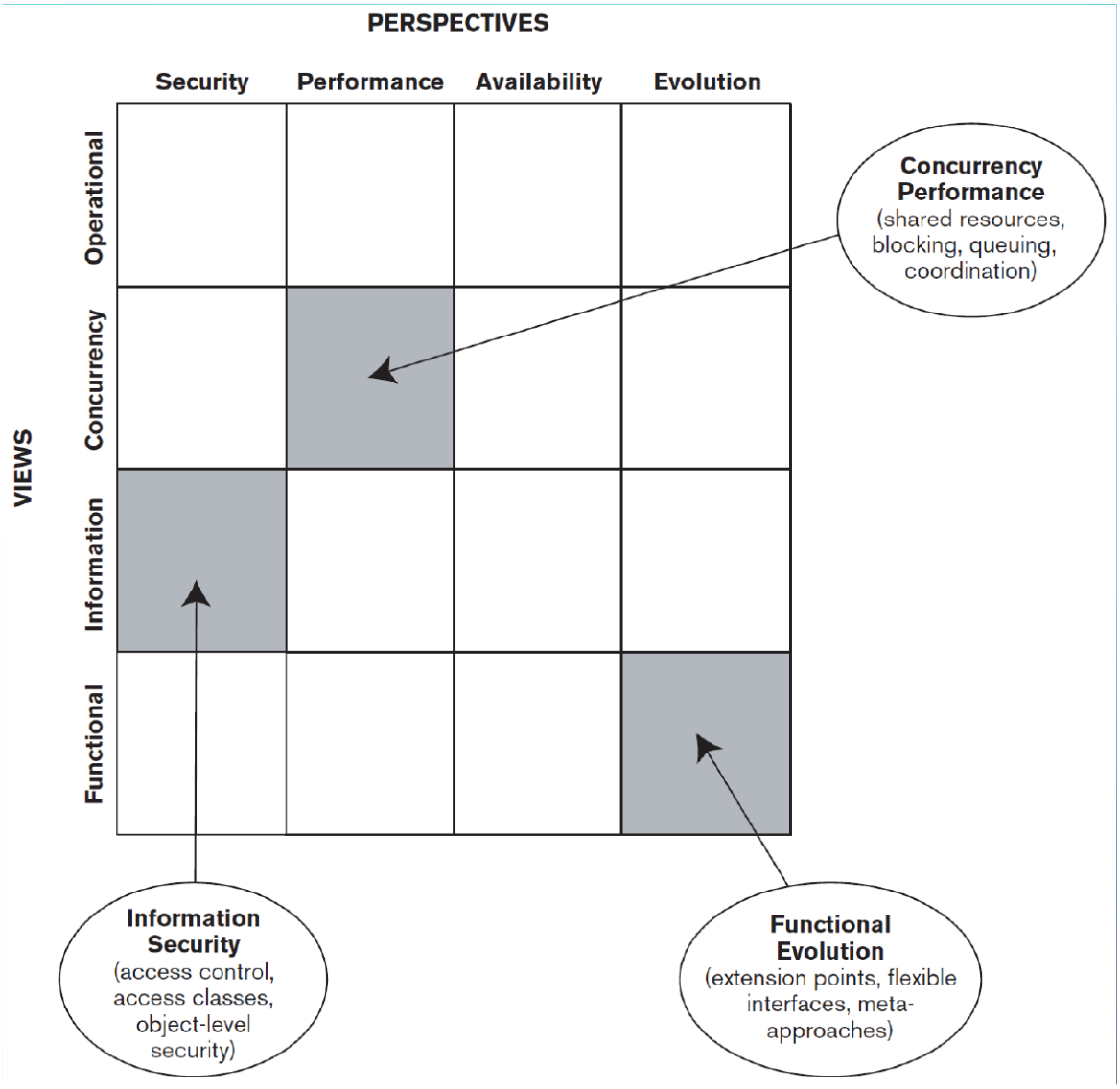


1. **PRIMARY PERSPECTIVE CATALOG**

Security – access to system resources
Performance and scalability – meeting performance and increased load satisfactorily
Availability and resilience – ensuring systems availability and coping with errors when they occur
Evolution – ensuring that the system can cope with likely changes

2. OTHER PERSPECTIVES

Accessibility – system to be used by people with disabilities
Development resource – system to be built within known constraints (cost, time, quality, people)
Internationalization – ability to be used independently by any language, culture or country
Location – ability to overcome problems brought by physical location between system's components
Regulation – ability to conform to regulations and laws
Usability – the ease with which people can interact with system effectively

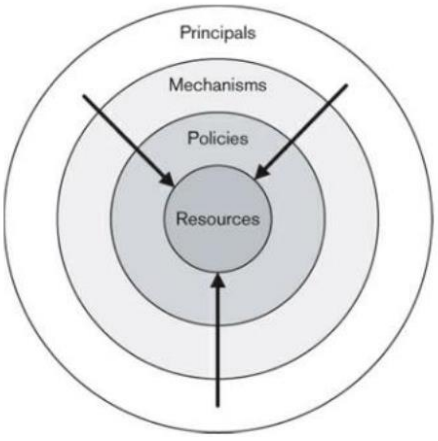


2. SECURITY perspective applicability to views

View	Applicability
Context	Identify external connections and threats to them
Functional	Identify security critical functional elements. Actual functional scope can be impacted by security needs
Information	What data needs to be protected
Concurrency	Indicate isolation on process level
Development	Guidelines, constraints and security policies for developers to be aware of
Deployment	Can contain security oriented hardware or software components
Operational	Definition of security principles and actions for clear responsibilities

1. CONCERNS OF SECURITY PERSPECTIVE

- **Confidentiality** is normally defined as limiting the disclosure of secrets to those who are legitimately allowed to access them.
  - Can be achieved using *access control*.
- **Integrity** is a guarantee that information cannot be changed undetectably
  - Can be achieved by "signing" data cryptographically.
- **Availability** – ensuring that potential attackers of your system cannot block its availability with denial-of-service attacks
- **Accountability** is the means of ensuring that every action can be unambiguously traced back to the principal who performed it
  - Can be achieved by *auditing*
- **Security treats**
  - Common types of security treats:
    - Physical layer attacks (Row hammer)
    - Unauthorized access and use (Pharming, spear phishing)
    - Software theft (License, software copy, code copy)
    - Information theft (Trojan, keylogging, phishing)
    - System failure (SQL Injection, CSRF)
    - Denial of service (Distributed denial of service attacks, slow-loris)
    - Software misuse (Unintentional software misuse to put system in faulted state or any above)
- **Resource guarding** – resources are guarded by policies which enforces technical mechanisms for principles to access resources
- **Security mechanism**
  - **Concerns and solutions for those security mechanism concerns:**
    - Authentication, authorization, auditing (SSO, Pass/username)
    - Information privacy and integrity (SSL/TLS)
    - Claiming of who you are not - Non-repudiation (Cryptography, message signing)
    - System availability (resilience to DOS attacks)
    - Security monitoring (Intrusion/misuse detection)



3. PERFORMANCE & SCALABILITY perspective applicability to views

View	Applicability
Functional	Identify and consolidate performance sensitive architectural elements
Information	Might require to consider replication or distribution of data to support goals
Concurrency	Performance and scalability can be tightly coupled with the architectural decisions in this view
Development	Guidelines and principles to support goals
Deployment	Clusters, grids, high-performance hardware etc.
Operational	Highlight needs for performance monitoring and management capabilities

4. AVAILABILITY & RESILIENCE perspective applicability to views

View	Applicability
Functional	Functional changes might be required to support different operation modes (occasionally connected client)
Information	Set of processes and systems for backup and recovery
Concurrency	Hardware replication or failover might affect concurrency model
Development	Design constraints via principles (all services should be startable, pausable and stoppable)
Deployment	Fault-tolerant prod environment, special software for clustering
Operational	Processes and mechanisms to identify and recover from problems

5. EVOLUTION perspective applicability to views

View	Applicability
Functional	If the evolution required is significant, the functional structure will need to reflect this
Information	If information evolution is needed, a flexible information model will be required
Concurrency	Evolutionary needs may dictate particular element packaging or some constraints on the concurrency structure (e.g., that it must be very simple)
Development	Evolution requirements may have a significant impact on the development environment that needs to be defined (e.g., enforcing portability guidelines)
Deployment	Rarely significant impact
Operational	Rarely significant impact

1. CONCERNS OF EVOLUTION PERSPECTIVE

• Product management (dedicated role)
• Magnitude of change (good software design, low coupling)
• Dimensions of change (software, hardware)
• Likelihood of change (requirement ambiguity)
• Timescale for change (how soon?)
• When to pay for change (ASAP or ALAP)
• Development complexity (meaningful design for extensibility)
• Preservation of knowledge (documentation, people)
• Reliability of changes (automated tests)

2. WHAT INVOKES CHANGE IN EVOLUTION PERSPECTIVE

•Misunderstood requirements
•Business change
•End user requirements
•Flaws of design or poor flexibility

3. DESIGN TACTICS FOR EVOLUTION PERSPECTIVE

•Separation of concerns
•Encapsulation
•Single point of definition – DRY (Do not Repeat Yourself)
•Functional cohesion
•Low coupling
•Abstract common services
•Abstraction and layering
•Generalization patterns
•Inversion of control / dependency injection
•Interface segregation