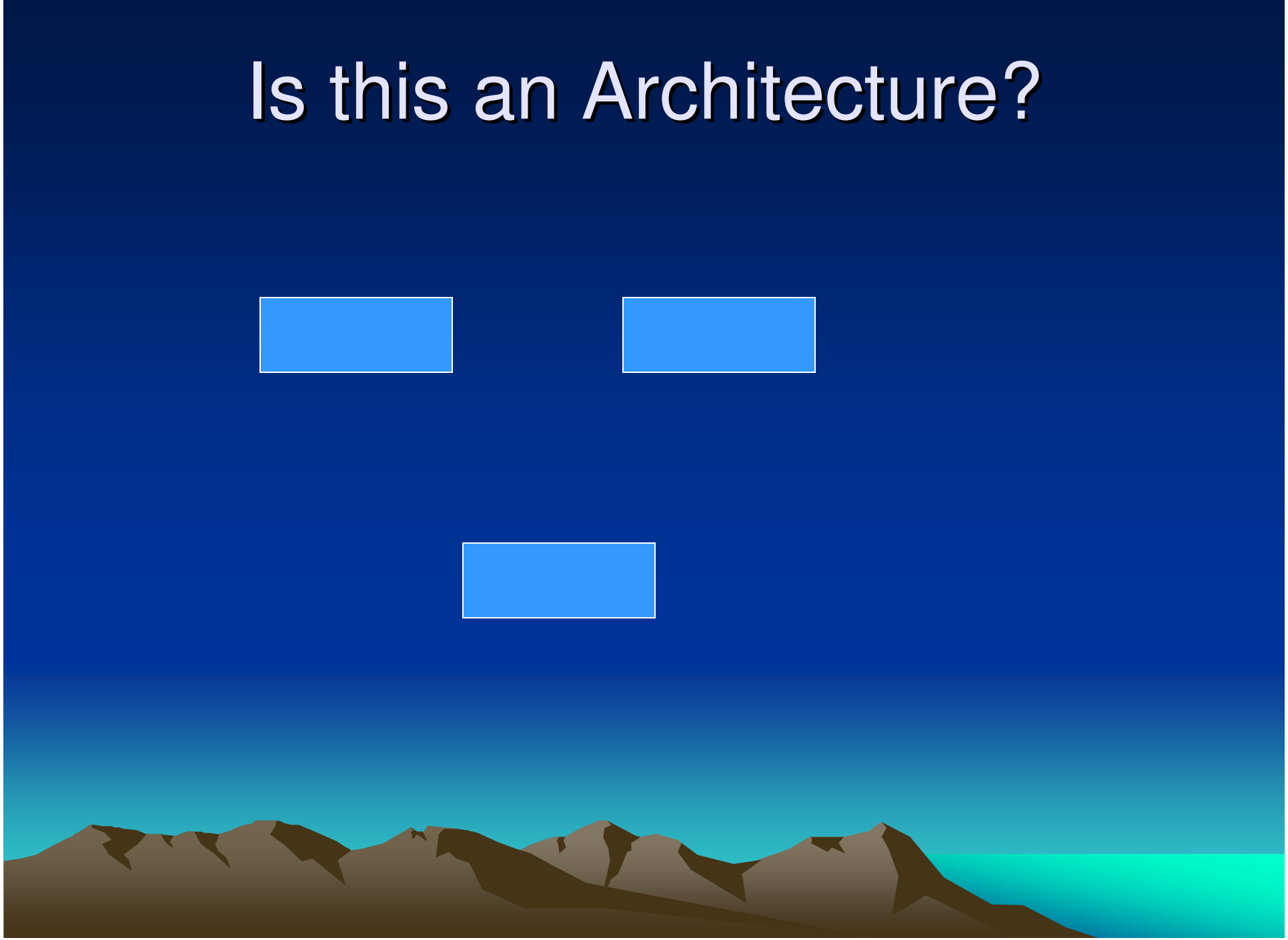
Software Architecture

Prof. R K Joshi

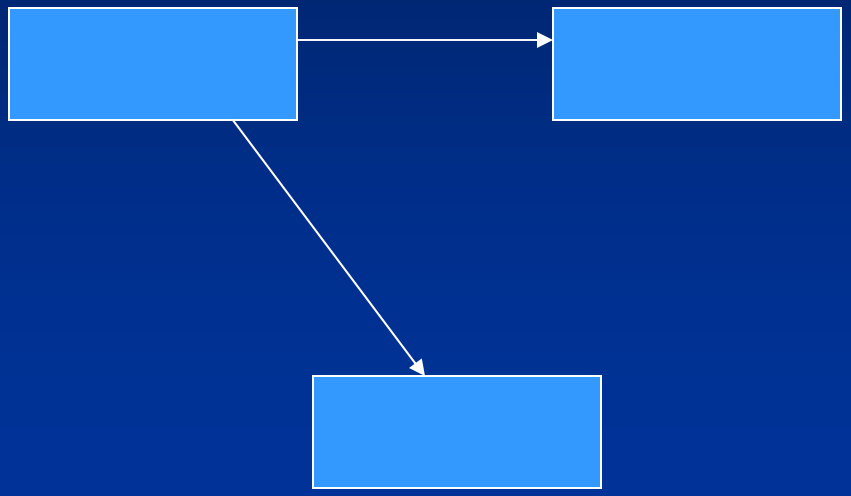
Department of Computer Science and Engineering
IIT Bombay

What is Architecture?
Software Architecture?

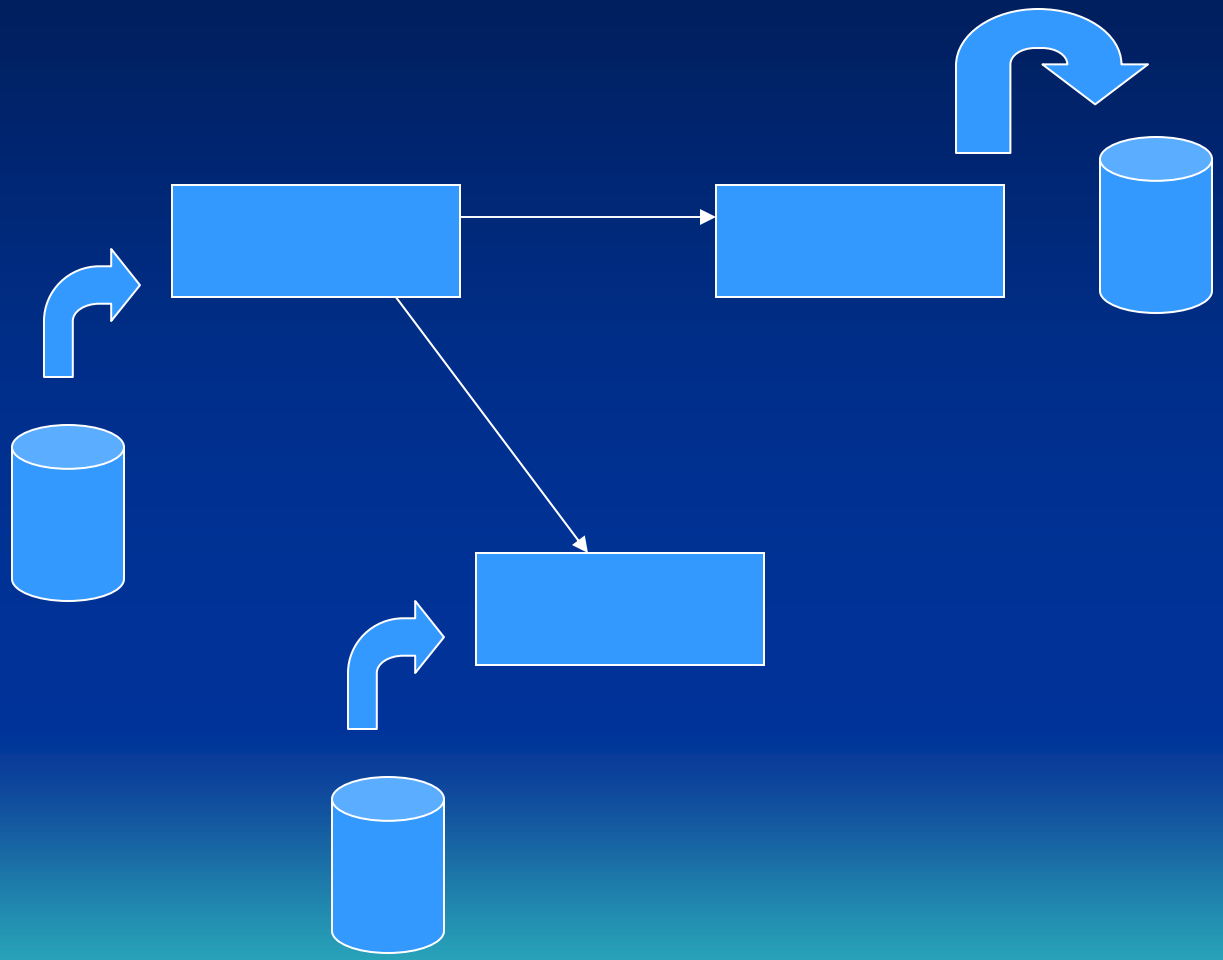
Is this an Architecture?



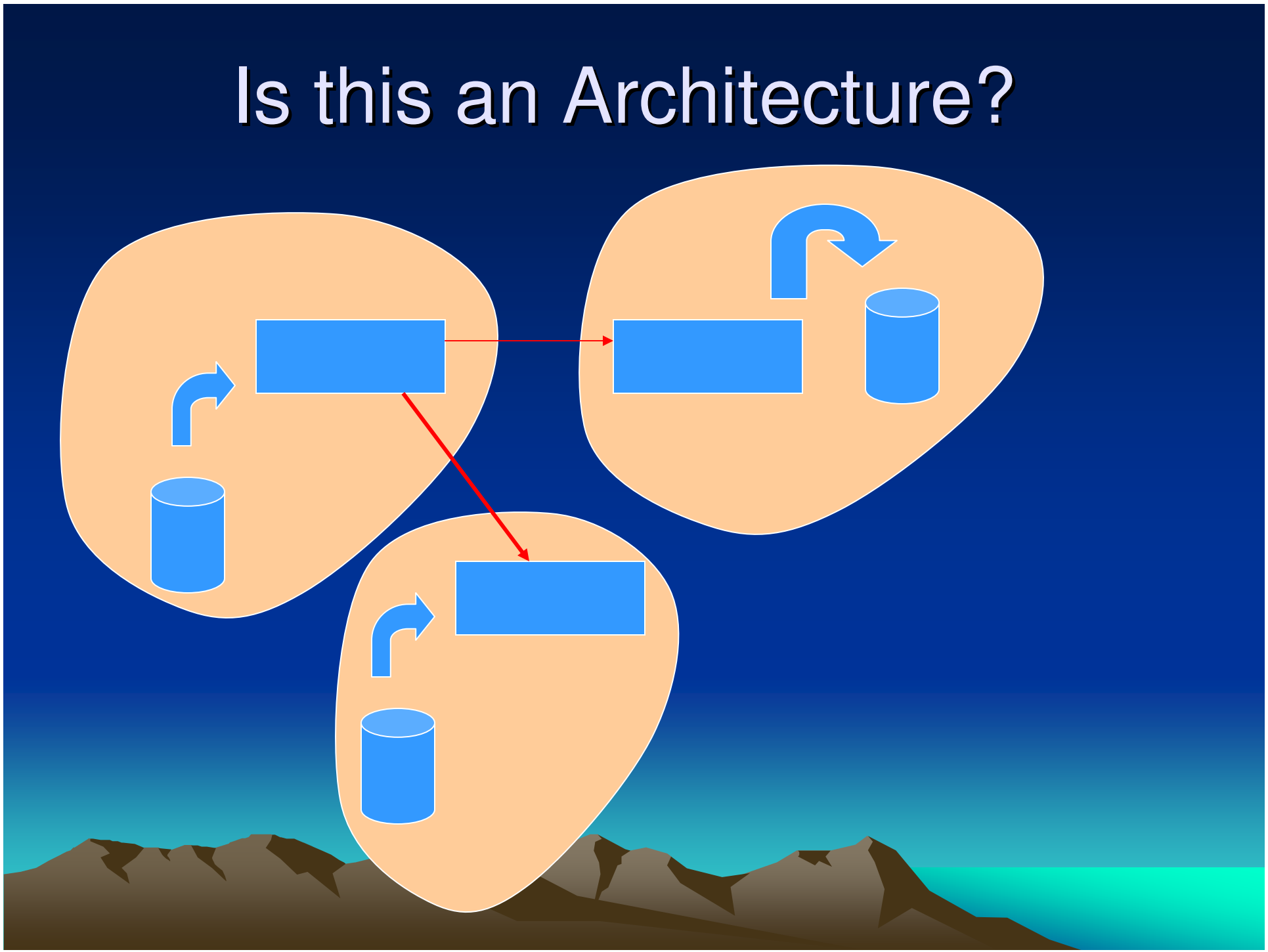
Is this an Architecture?



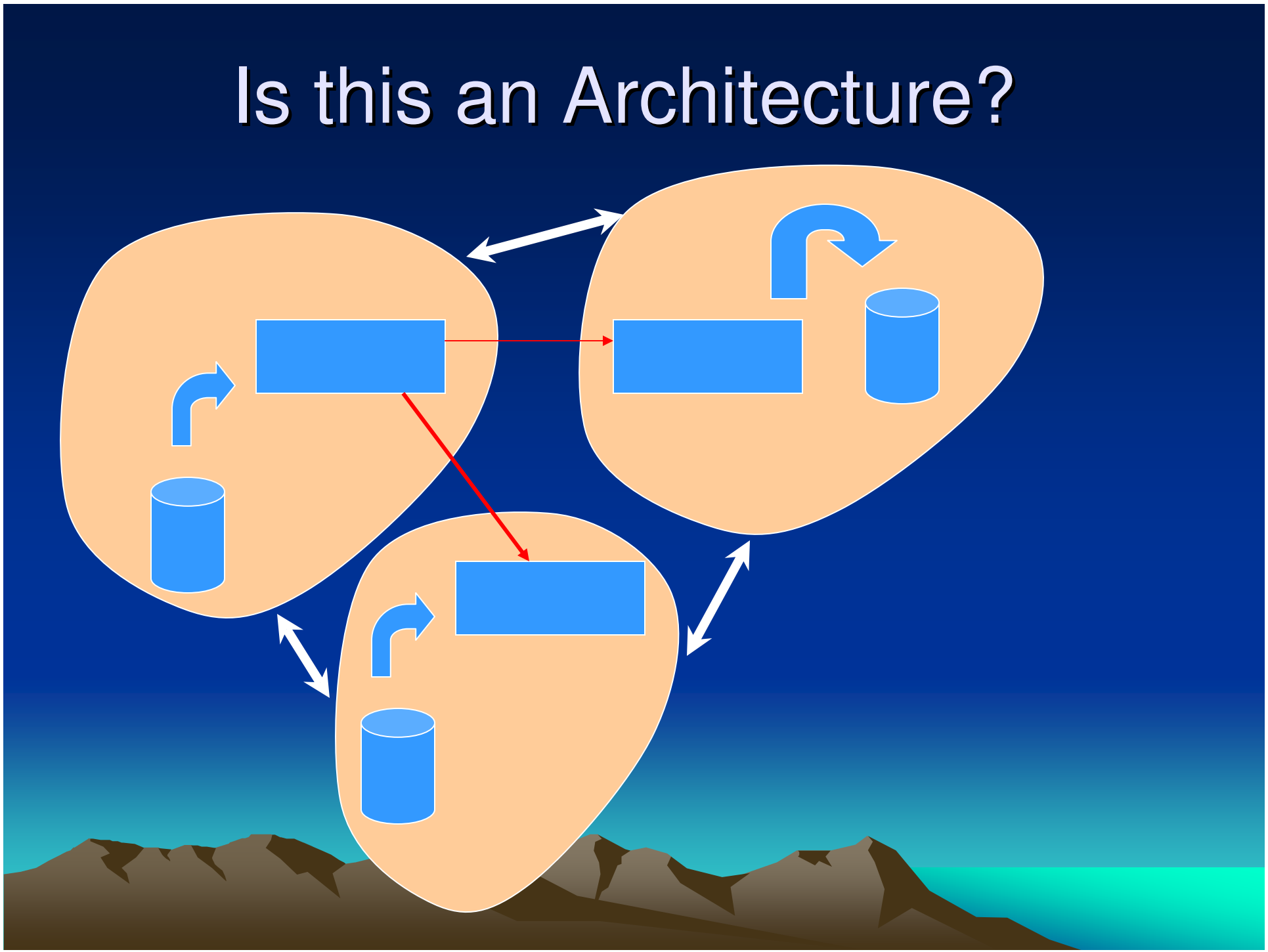
Is this an Architecture?



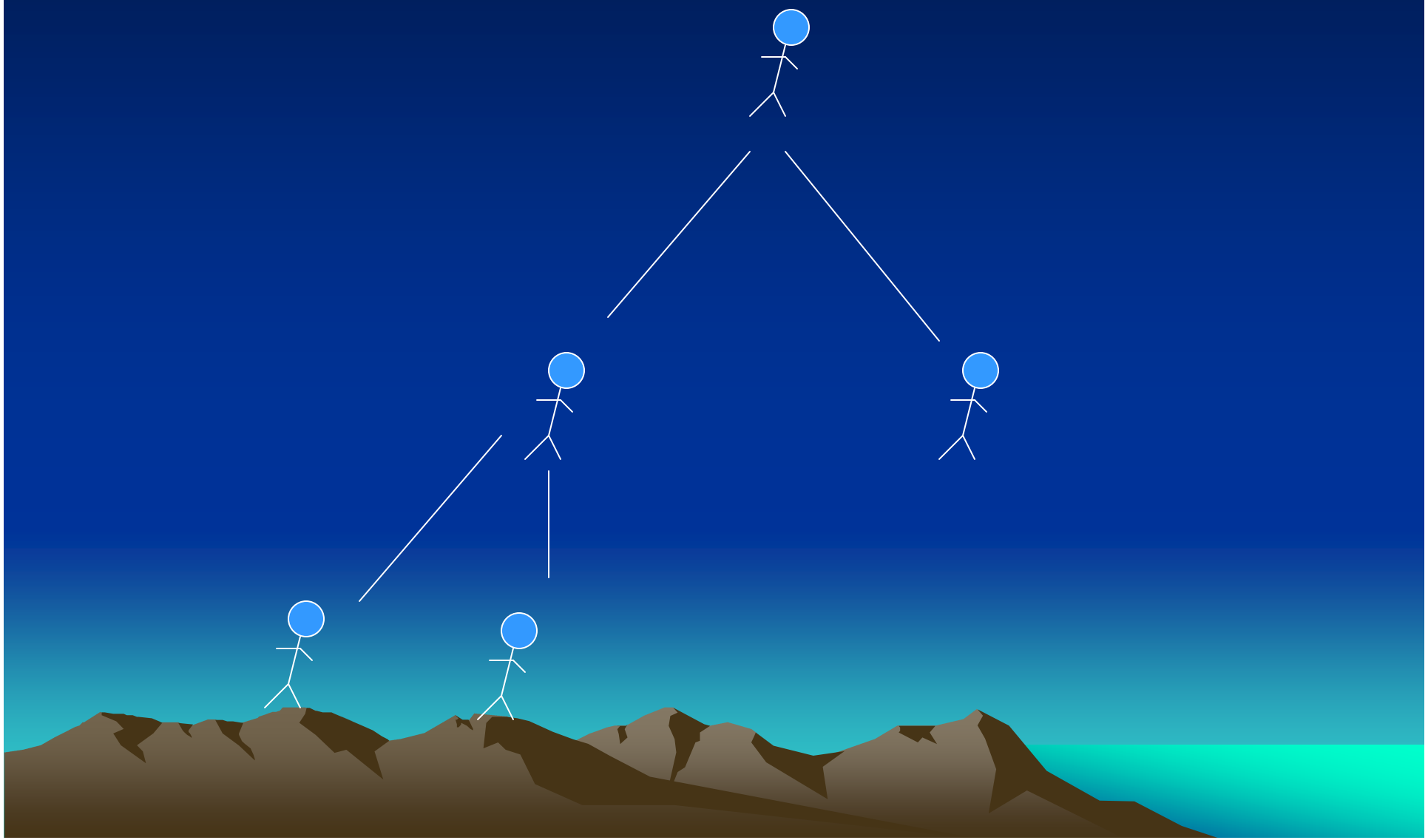
Is this an Architecture?



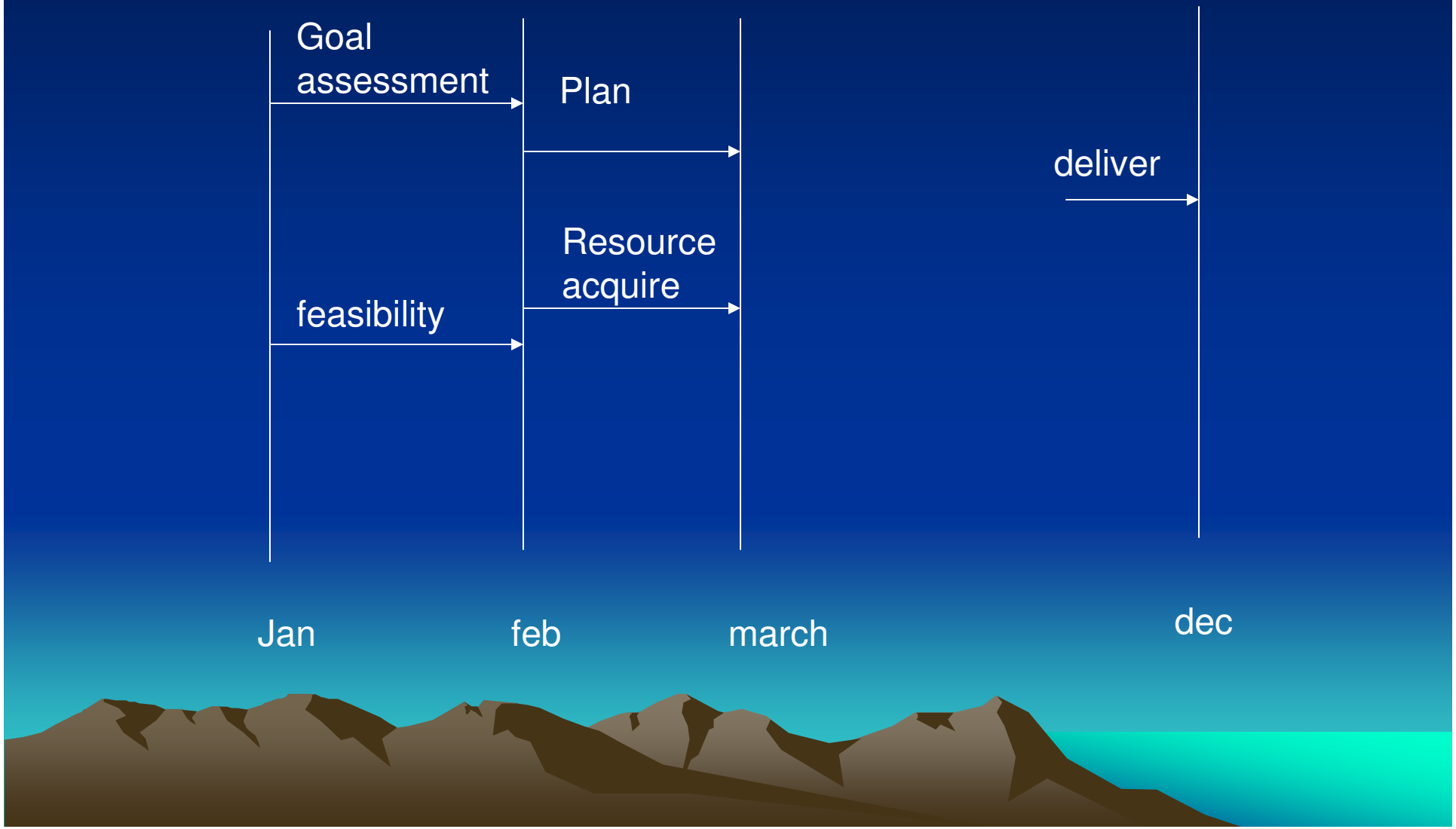
Is this an Architecture?



Is this an Architecture?



Is this an architecture?



Why Do We need Architecture?

- Understand the system
  - Complex systems
- Organize the development
  - According to architectural partitioning
- Reuse
  - Componentization
- Evolution
  - Changes and dependencies

Processes and Products

- Process Architecture
  - About the process of software development
- Product Architecture
  - About the product under development

Several Approaches to Architecture

[e.g. see in Malveau & Moubray 2001]

- Zachman Framework (IBM)
  - 30 architectural viewpoints
- Open Distributed Processing (ISO standard)
  - 5 viewpoint reference model
- Domain Analysis
- 4+1 View Model (Unified Process-Rational)
- Academic Software Architecture

The Zachman Framework

[Zachman institute of framework advancement]

- "To keep the business from disintegrating, the concept of information systems architecture is becoming less of an option and more of a necessity" – Zachman in 1987
- Developed by Zachman from observing how architectures in engineering, construction and manufacturing managed *change*
- Intersection between roles in the design process and product abstractions
- Roles (in rows): Owner, Designer, Builder
- Product Abstractions (in columns): What is it made up of (data), How it works (process), Where are the components located (geometry)
- 3 additional columns: Who (people), When (time), Why (motivation)
- 2 additional rows: Planner, Subcontractor
- Columns have no order

Zachman Row 1: Planner's view Things important to business

- Column *why* (Motivation)
  - Business motivation, End: goals and measures of each goal, defines scope and boundaries
- Column *How* (Process)
  - Class of High level processes with inputs and output
- Column *What* (Data)
  - Class of Business data objects that enterprise is interested in
- Column Who (People)
  - Class of business organizations/people
- Column When (Time)
  - Events related to each process
- Column Where (Network)
  - Class of locations where the processes are executed

Zachman Row 2: Owner's view Enterprise Semantic model

- Column *why* (Motivation)
  - Ends/business goals and means/business strategy
- Column *How* (Process)
  - E.g. structured method-processes and flows
- Column *What* (Data)
  - E/R type model representing business entities, business relationships
- Column Who (People)
  - Work flow model- control, coordination, operation
- Column When (Time)
  - Master schedule (e.g. PERT)
- Column Where (Network)
  - Nodes, branches, warehouses etc.

Zachman Rows 3,4,5

- Designer
  - Logical models-e.g. data entity relations, ooad
- Builder
  - Physical models-e.g. Tables, hardware
- Subcontractor/Implementor
  - Filelds, control blocks, statements

Open Distributed Processing Reference Model

- For architecture supporting distribution, internetworking, interoperability and portability
- Five viewpoints
  - Enterprise (purpose, scope and policies)
  - Information (semantics of information and information processing)
  - Computational (functional decomposition)
  - Engineering (infrastructure to support distribution)
  - Technology (for implementation: Mappings between objects and specific standards and technologies)
- The set of viewpoints is not closed
- Each of the viewpoint is object oriented

ODP: Enterprise viewpoint

- Directly understandable by managers and end users
- Defines business purpose, scope and policies
- Includes permissions, prohibitions and obligations
- Example:
  - Active objects: managers, tellers, customers
  - Passive objects: accounts
  - Bank managers must advise customers when interest rate changes (obligation)
  - Cash less than 40000 can be drawn per day (prohibition)
  - Money can be deposited (permission)

ODP: Information viewpoint

- Definitions of information schemas as objects
  - State and structure of objects
    - E.g. account = balance and amount withdrawn today
- Three kinds of schemas:
  - static
    - At midnight, amount withdrawn today=2000
  - Invariant
    - At anytime, amount withdrawn today <=40000
  - Dynamic
    - A deposit of X increases the balance by X and a withdrawal of X decreases the balance by X
    - Always constrained by invariant
- Schemas may relate objects
  - E.g. in customer object: *owns account* static schema

ODP: Computational viewpoint

- Software components which are capable of supporting distribution
- Large grained object encapsulations, subsystem interfaces and behaviors
- Objects can offer multiple interfaces
- 3 types of interactions among objects
  - Operational : client-server, RPC : with parameters and results
  - Stream oriented
  - Signal oriented
- Inheritance of Interface and subtyping
- Operations such as object creation, trading for an interface, interface creation, binding, operation invocation
- Examples
  - Application objects: Bank branch with bank teller (deposit, withdraw) and bank manager (create account, deposit, withdraw) interfaces for customers
  - ODP infrastructural objects: Trader

ODP: Engineering viewpoint

- Brings out the distributed nature of the system
- Objects and Channels
- Objects
  - Basic engineering objects correspond to computational objects
  - Infrastructural objects such as protocol objects
    - E.g. stub, binder and protocol object (proxy/skeletons) + communication interface between protocol objects
- Engineering structure of the system is described
  - E.g. cluster, nucleus object, capsule of clusters, a machine node, a cluster may contain many engineering objects, an object can contain many activities, inter-cluster communication via channels

ODP: Transparencies Defined

- Access
  - hides the difference in data representation and invocation mechanism – enables heterogeneous systems to communicate
- Failure
  - Hides failures and possible recoveries of objects for fault tolerance
- Location
  - Hides the location information while finding and bind to an object
- Relocation
  - Masks the changes in the location of an object from its clients
- Migration
  - Masks the awareness of changes in location of the object from itself and from others
- Replication
  - Masks the existence of replicated objects
- Persistence
  - Masks activation and deactivation of objects
- Transaction
  - Masks coordination of activities to achieve consistency

4+1 View Model

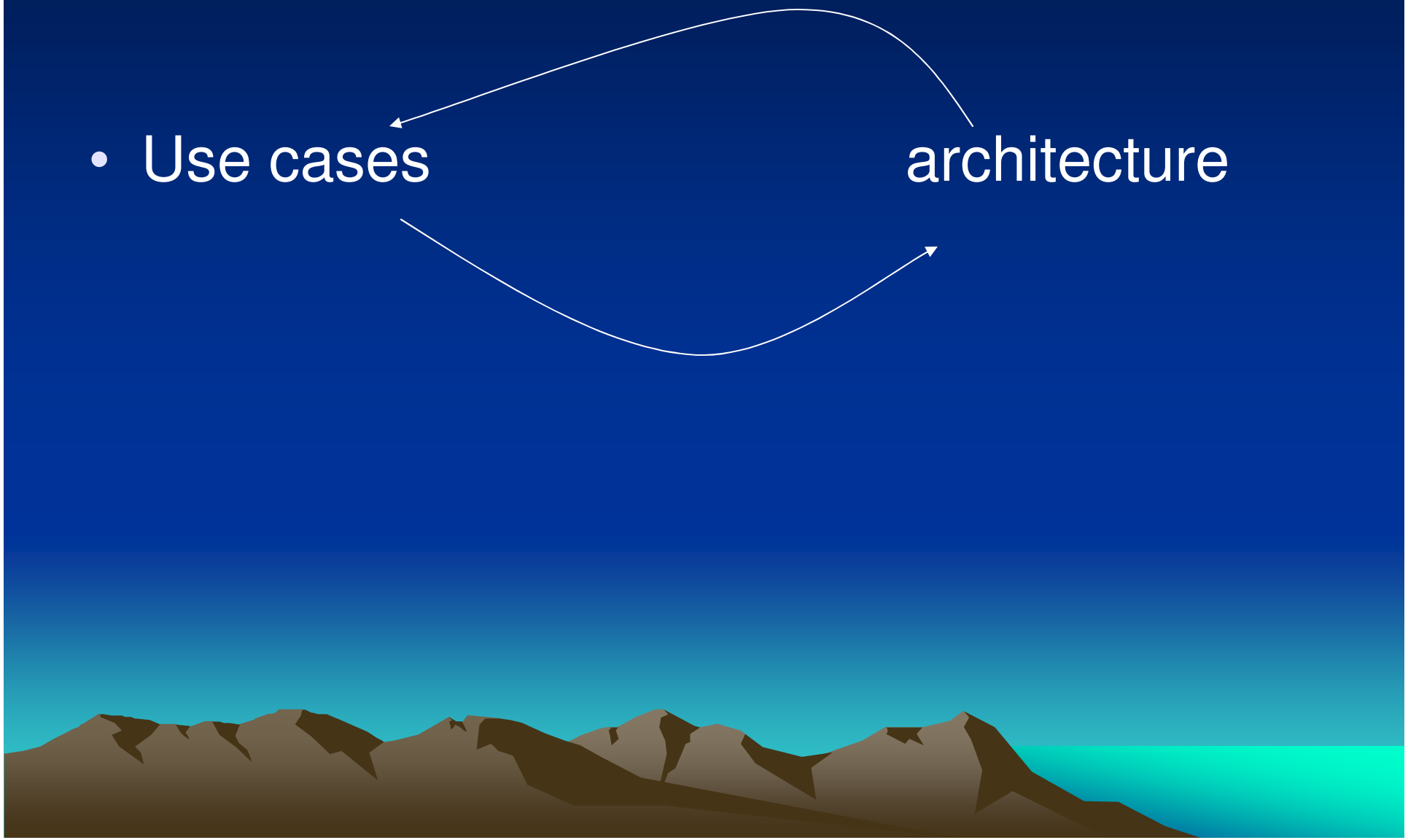
[P.B. Kutchen, 1995]

- Sometimes software architecture suffers from system designers who go too far..other software engineers fail to address the concerns of all customers
- 4+1 view model: Has 5 concurrent views
- Logical view- e.g. object model using object oriented design method
- Process view – concurrency and synchronization aspects
- Physical view – mapping of components to hardware, distribution aspect
- Development view – organization of the actual software modules – libraries, packages, subsystems
- Use case view

Unified Process Model of Architecture

- Architecture description is a proper extract of the models of the system (use case model, analysis model, design model, deployment model, implementation model)
  - e.g. Contains only architecturally significant use cases, whereas final use case model contains all use cases;
  - Similarly architectural view of design model realizes only the architectural use cases
  - First version of architecture is extract at the end of elaboration phase and so on
- Developed iteratively during elaboration phase
- Focus on significant structural elements of the system
  - Subsystems, classes, components, nodes
- Use cases and architecture

Chicken and Egg



Commonly occurring Architectural Patterns

- Fundamental structural organization schemas
- For example:
  - Layers
  - Pipes and Filters
  - Blackboard
  - Broker
  - Model-View-Controller
  - Presentation-Abstraction-Control
  - Microkernel
  - Reflection

Frameworks: An Approach to Architecture

- Partially complete software
- It is instantiated as a product
- For product families/product lines
- Frozen spots and hot spots

Enabling Techniques

- Abstraction
- Encapsulation
- Information Hiding
- Modularization
- Separation of Concerns
- Coupling and Cohesion
- Sufficiency, Completeness and Primitiveness
- Separation of Policy and Implementation
- Separation of Interface and Implementation
- Single point of reference
- Divide and Conquer

References/Readings

- John Zachman, *A Framework for Information Systems Architecture",* **IBM Systems Journal**, Vol 26, No 3, 1987
- Kerry Raymond, *Reference model for Open Distributed Processing (RM-ODP): Introduction*, CRC for Distributed Systems Technology, University of Queensland
- Raphel Malveau, Thomas Mowbray, Software Architect Bootcamp, Prentice Hall 2001
- Buschmann, Meuneir, Rohnert, Sommerlad, Stal, Pattern-oriented Software Architecture: A system of patterns, John Wiley & Sons, 1996
- P.B. Kruchten, The 4+1 View Architecture, **IEEE Software**, November 1995
- Jacobson, Booch, Rumbaugh, The Unified Software Development Process, Addison Wesley Longman, 1999