VIEWS OF VIEWPOINTS

1. VIEWPOINTS

1.1. GENERAL INFORMATION

A VIEW IS A REPRESENTATION OF ONE OR MORE STRUCTURAL ASPECTS OF AN ARCHITECTURE THAT ILLUSTRATES HOW THE ARCHITECTURE ADDRESSES ONE OR MORE CONCERNS HELD BY ONE OR MORE OF ITS STAKEHOLDERS

A VIEWPOINT IS A COLLECTION OF PATTERNS, TEMPLATES, AND CONVENTIONS FOR CONSTRUCTING ONE TYPE OF VIEW. IT DEFINES THE STAKEHOLDERS WHOSE CONCERNS ARE REFLECTED IN THE VIEWPOINT AND THE GUIDELINES, PRINCIPLES, AND TEMPLATE MODELS FOR CONSTRUCTING ITS VIEWS.

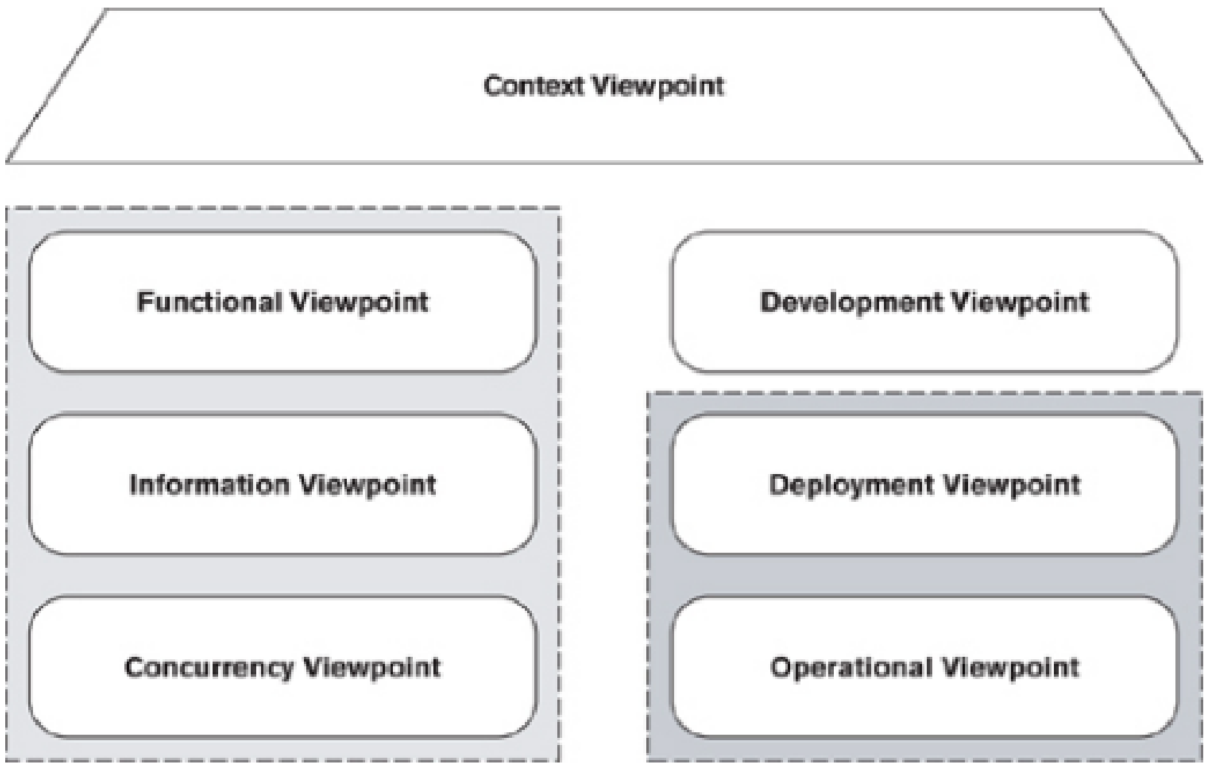
1.2. BENEFITS OF USING VIEWPOINTS:

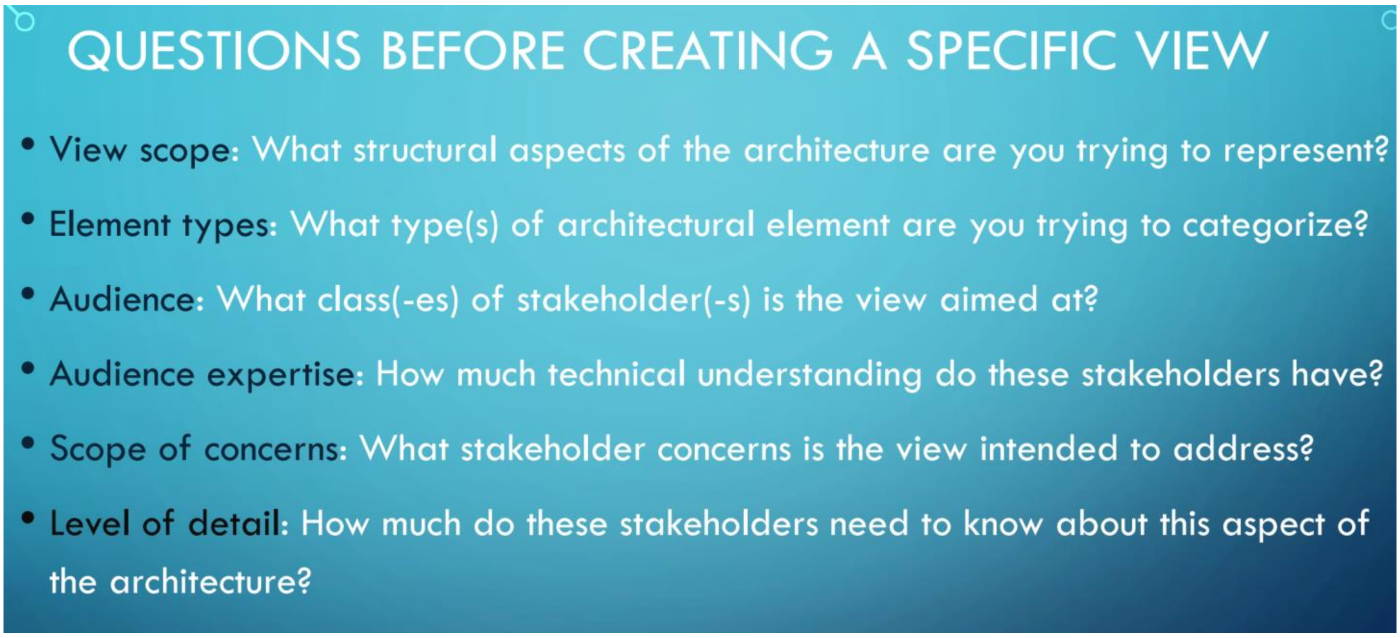
• Separation of concerns
• Targeted communication with stakeholder groups
• Management of complexity
• Improved developer focus

1.3. PITFALLS OF USING VIEWPOINTS:

• Inconsistency
• Selection of wrong set of views
• Fragmentation

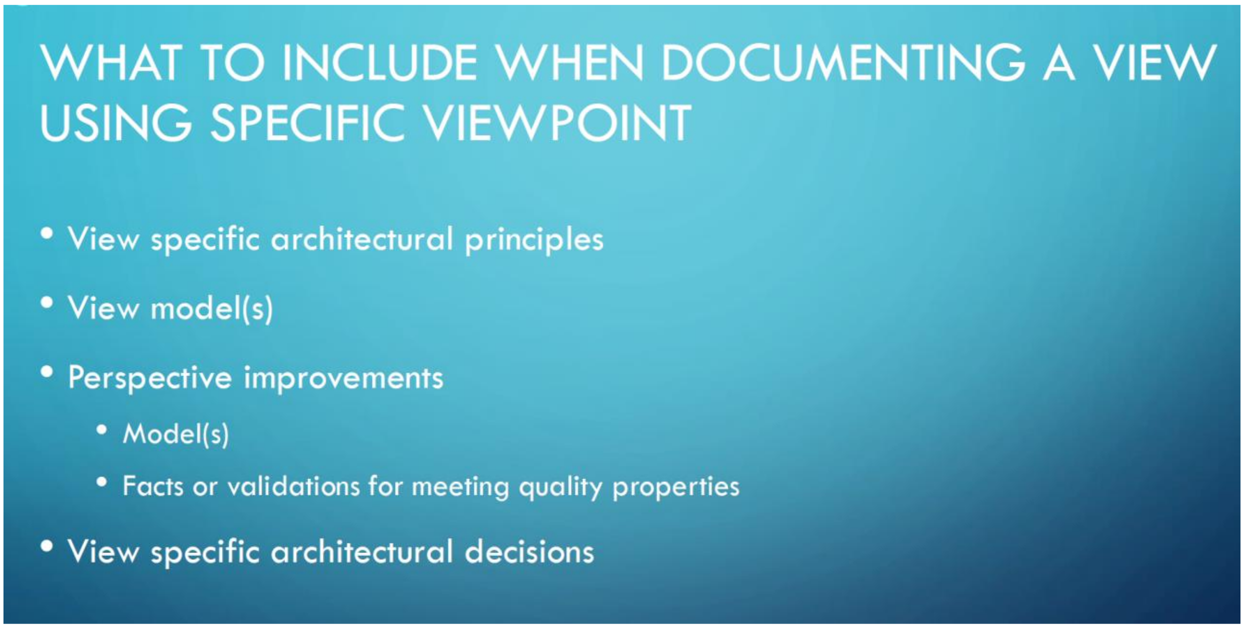
1.4. VIEWPOINT CATALOG:





QUESTIONS BEFORE CREATING A SPECIFIC VIEW

- View scope: What structural aspects of the architecture are you trying to represent?
- Element types: What type(s) of architectural element are you trying to categorize?
- Audience: What class(-es) of stakeholder(-s) is the view aimed at?
- Audience expertise: How much technical understanding do these stakeholders have?
- Scope of concerns: What stakeholder concerns is the view intended to address?
- Level of detail: How much do these stakeholders need to know about this aspect of the architecture?

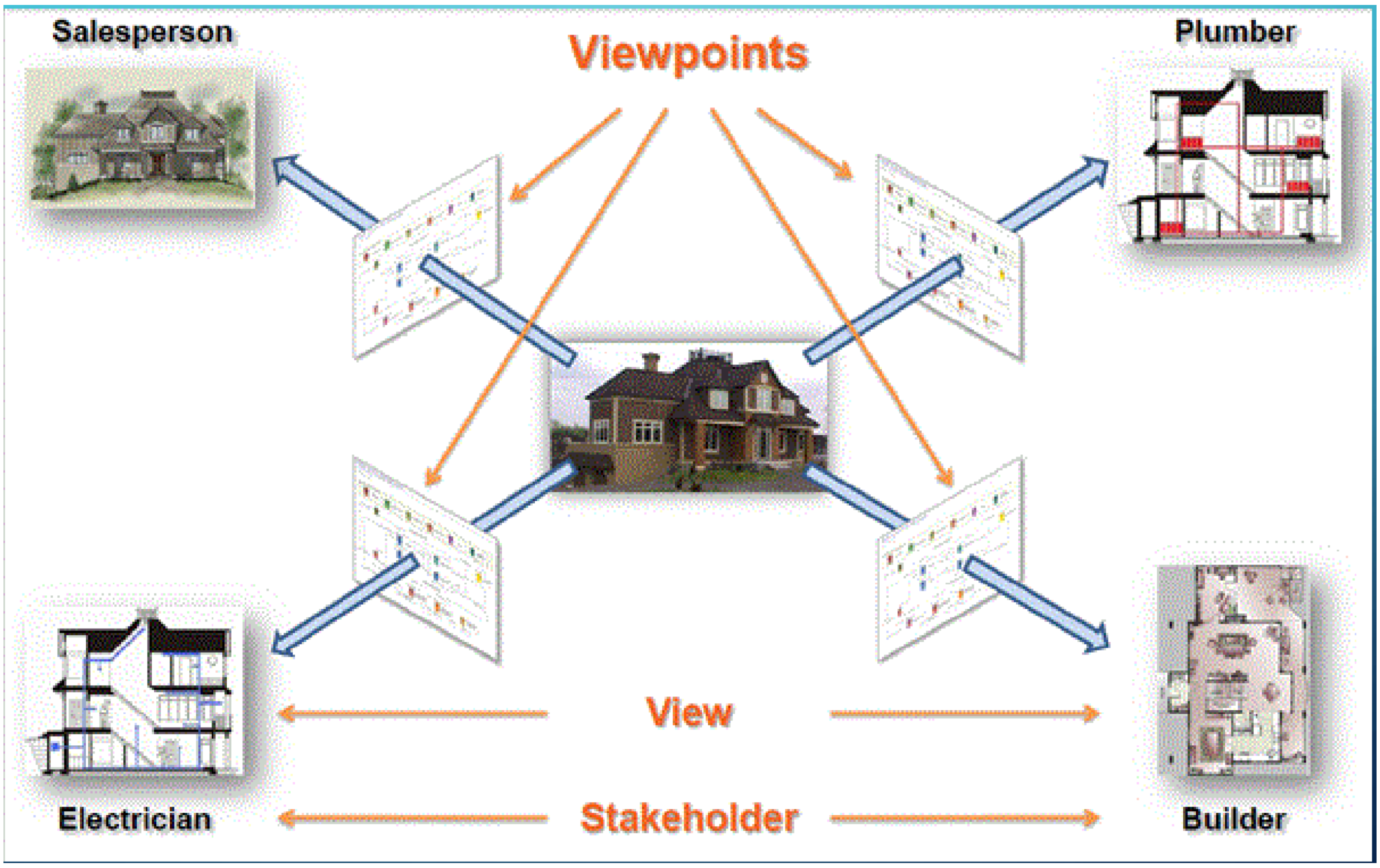


WHAT TO INCLUDE WHEN DOCUMENTING A VIEW USING SPECIFIC VIEWPOINT

- View specific architectural principles
- View model(s)
- Perspective improvements
  - Model(s)
  - Facts or validations for meeting quality properties
- View specific architectural decisions

1.5. VIEWPOINTS SUMMARY:

• The **Context viewpoint** describes the relationships, dependencies, and interactions between the system and its environment (the people, systems, and external entities with which it interacts)
• The **Functional**, **Information** and **Concurrency** viewpoints characterize the fundamental organization of the system
• The **Development viewpoint** exists to support the system's construction
• The **Deployment** and **Operational viewpoints** characterize the system once in its live environment.



1.6. SHORT DESCRIPTION OF VIEWPOINTS

✓ **CONTEXT VIEWPOINT** – DESCRIBES THE RELATIONSHIPS, DEPENDENCIES AND INTERACTIONS BETWEEN SYSTEM AND ITS ENVIRONMENT (THE PEOPLE, SYSTEMS AND EXTERNAL ENTITIES WITH WHICH IT INTERACTS)

✓ **INFORMATION VIEWPOINT** – DESCRIBES THE WAY THE SYSTEM STORES, MANIPULATES, MANAGES AND DISTRIBUTES INFORMATION

✓ **CONCURRENCY VIEWPOINT** – DESCRIBES THE CONCURRENCY STRUCTURE OF THE SYSTEM AND MAPS FUNCTIONAL ELEMENTS TO CONCURRENCY UNITS TO CLEARLY IDENTIFY THE PARTS OF THE SYSTEM THAT CAN EXECUTE CONCURRENTLY AND HOW THIS IS COORDINATED AND CONTROLLED

✓ **DEVELOPMENT VIEWPOINT** – DESCRIBES THE ARCHITECTURE THAT SUPPORTS THE SOFTWARE DEVELOPMENT PROCESS. COMMON OR KEY DESIGN PATTERNS (i.e. "Adapter", "Strategy" etc.) SHOULD BE INCLUDED IN THIS PART THAT TACKLE SIGNIFICANT DESIGN DECISIONS

✓ **OPERATIONAL VIEWPOINT** – DESCRIBES HOW THE SYSTEM WILL BE OPERATED, ADMINISTERED AND SUPPORTED WHEN RUNNING IN PRODUCTION ENVIRONMENT

1.7. SUGGESTED DIAGRAMS FOR VIEWS BY VIEWPOINTS:

Context View: Draw.io -> Software -> ArchiMate Diagram

Functional View: Draw.io -> Software -> Components Diagram

Informational View: Draw.io -> Software -> “Database 3” or “Entity-Relation” Diagram

Concurrency View:

Development View: Draw.io -> UML -> UML 2 or Business -> BPMN or Software -> Class 2

Deployment View: Draw.io -> Network -> AWS Diagram

Operational View: Text is preferred instead of diagrams

1.8. NOTES ON DIAGRAMS:

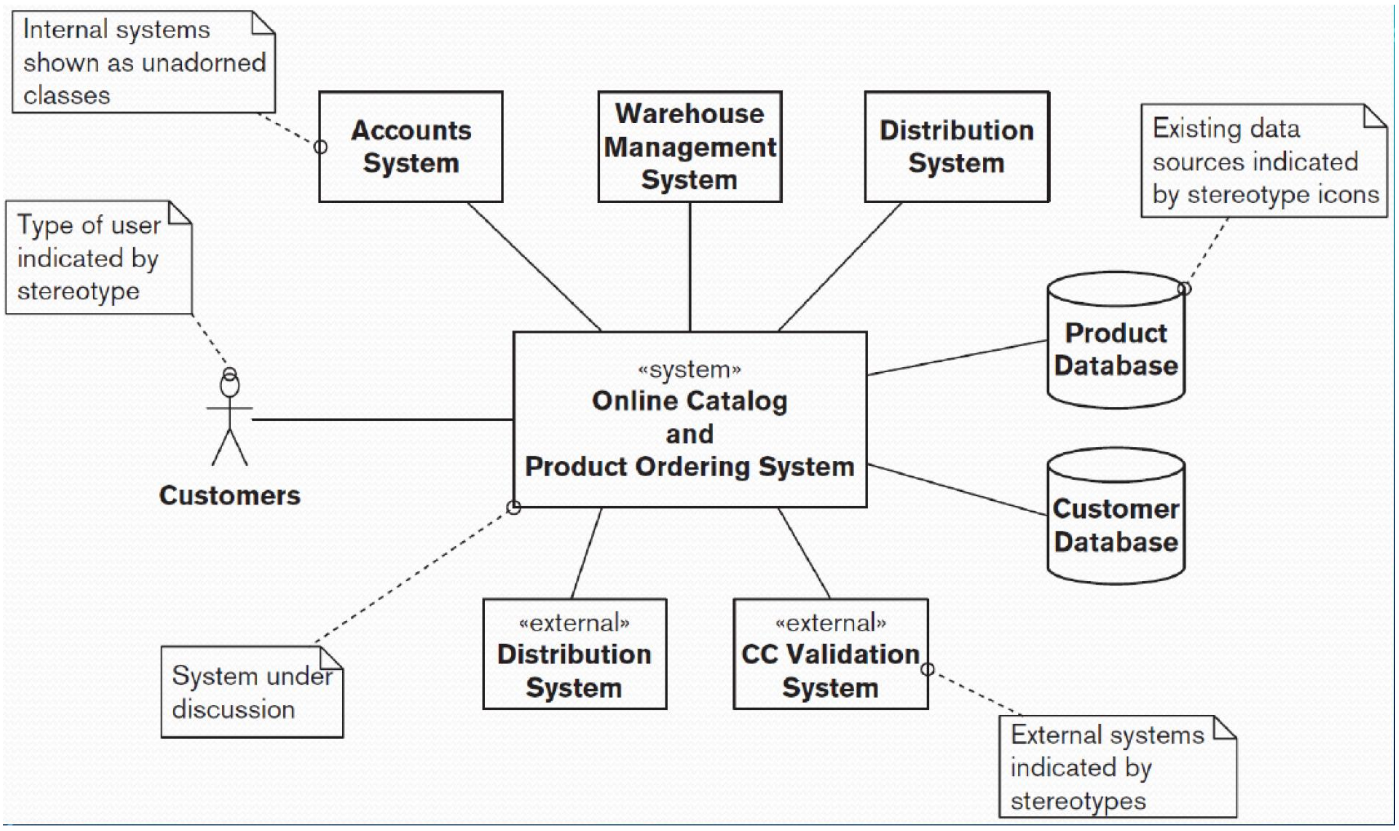
ArchiMate (originally from **Archi**tecture-Ani**mate**) is an open and independent **enterprise architecture modeling language** to support the description, analysis and visualization of architecture within and across business domains in an unambiguous way.

ArchiMate is a technical standard from The Open Group and is based on the concepts of the **IEEE 1471** standard. It is supported by various tool vendors and consulting firms. ArchiMate is also a registered trademark of The Open Group. The Open Group has a certification program for ArchiMate users, software tools and courses.

ArchiMate distinguishes itself from other languages such as Unified Modeling Language (UML) and Business Process Modeling and Notation (BPMN) by its enterprise modelling scope.

2. CONTEXT VIEWPOINT

2.1. ELEMENTS OF CONTEXT VIEW


Internal systems shown as unadorned classes
Accounts System
Warehouse Management System
Distribution System
Existing data sources indicated by stereotype icons
Type of user indicated by stereotype
Product Database
«system» Online Catalog and Product Ordering System
Customers
Customer Database
«external» Distribution System
«external» CC Validation System
System under discussion
External systems indicated by stereotypes

2.2. CONCERNS OF CONTEXT VIEWPOINT

• System scope and responsibilities
• System users
• Identity of external systems, services and data
• Characteristics of external interfaces

2.3. CONTEXT VIEWPOINT MODELS:

• Context model
• Interaction scenarios

2.4. CONTEXT VIEWPOINT STAKEHOLDERS:

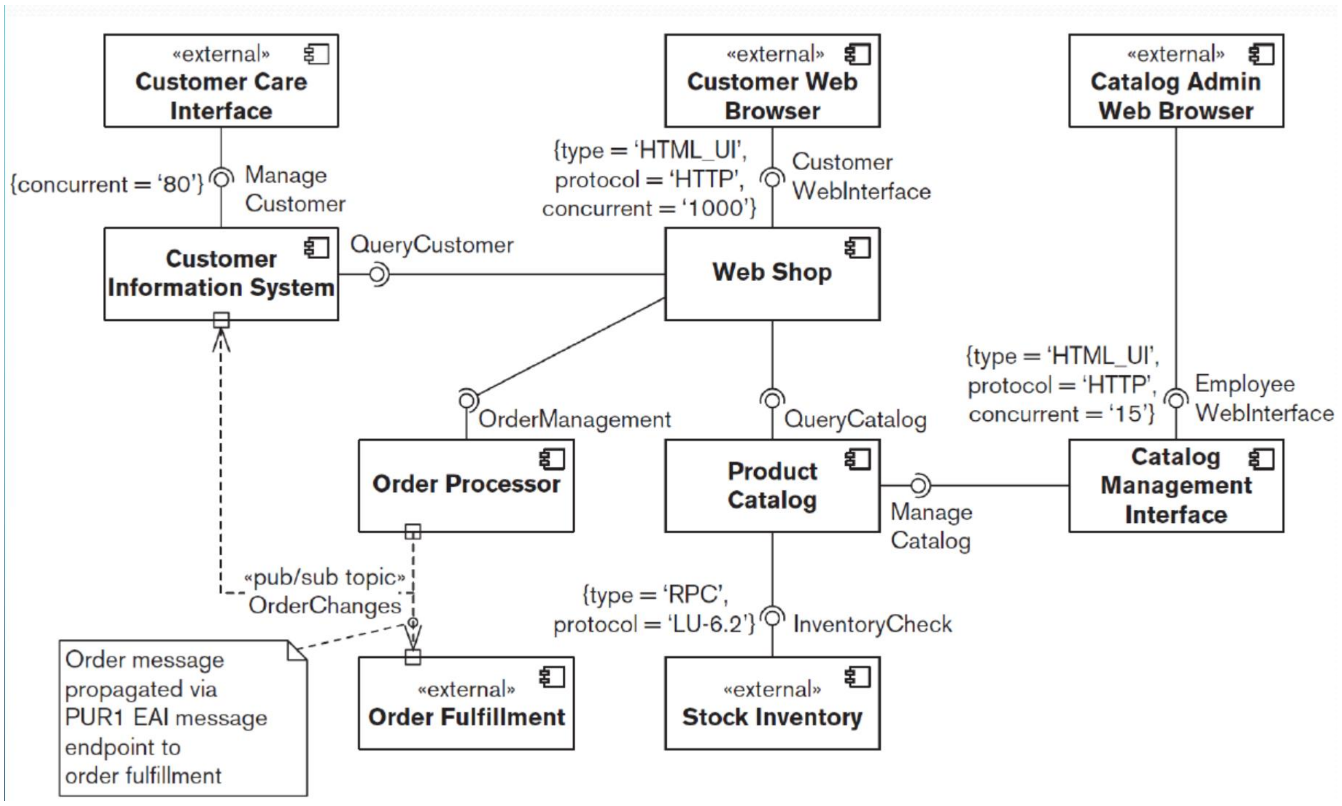
✓ All
✓ Especially
  - Acquirers
  - Users
  - Developers

2.5. CHECKLIST:

✓ Level of detail
✓ Completeness of external dependencies
✓ No use of jargon
✓ Added value and meaningfulness to stakeholders

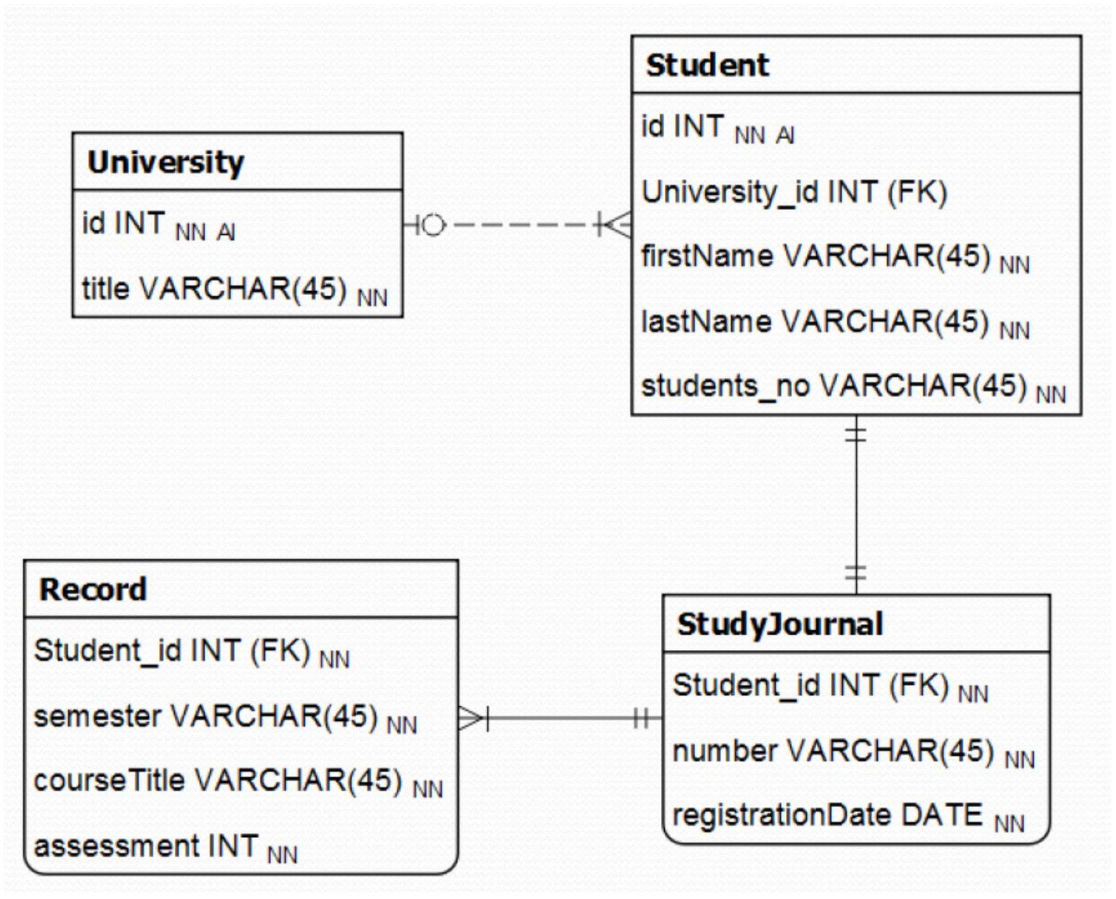
3. FUNCTIONAL VIEWPOINT

3.1. ELEMENTS OF FUNCTIONAL VIEW:


«external» Customer Care Interface
«external» Customer Web Browser
«external» Catalog Admin Web Browser
{concurrent = '80'}
Manage Customer
{type = 'HTML_UI', protocol = 'HTTP', concurrent = '1000'}
Customer WebInterface
Customer Information System
QueryCustomer
Web Shop
{type = 'HTML_UI', protocol = 'HTTP', concurrent = '15'}
Employee WebInterface
OrderManagement
QueryCatalog
Order Processor
Product Catalog
Catalog Management Interface
Manage Catalog
«pub/sub topic» OrderChanges
{type = 'RPC', protocol = 'LU-6.2'}
InventoryCheck
Order message propagated via PUR1 EAI message endpoint to order fulfillment
«external» Order Fulfillment
«external» Stock Inventory

4. INFORMATION VIEWPOINT

4.1. ELEMENTS OF INFORMATION VIEW

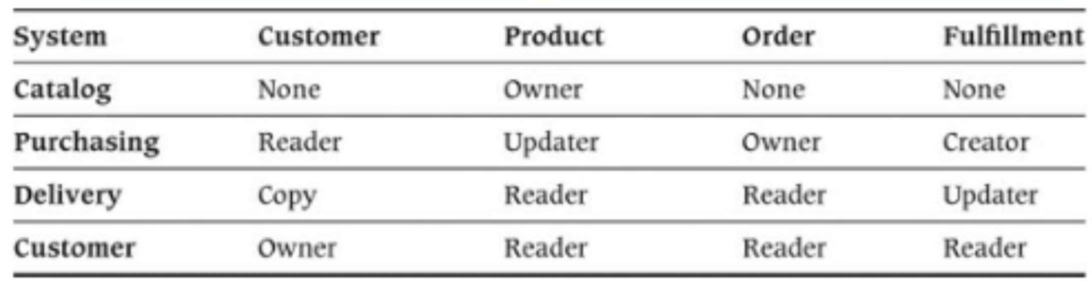


4.2. CONCERNS OF INFORMATION VIEW

• Information structure and content
• Information purpose and usage
• Information ownership
• Information flow
• Information storage models

4.3. MODELS OF INFORMATION VIEW

- Static information structure model
- Information flow models
- Information lifecycle models
- Information ownership models
    - I.e. Information ownership grid:



System	Customer	Product	Order	Fulfillment
Catalog	None	Owner	None	None
Purchasing	Reader	Updater	Owner	Creator
Delivery	Copy	Reader	Reader	Updater
Customer	Owner	Reader	Reader	Reader

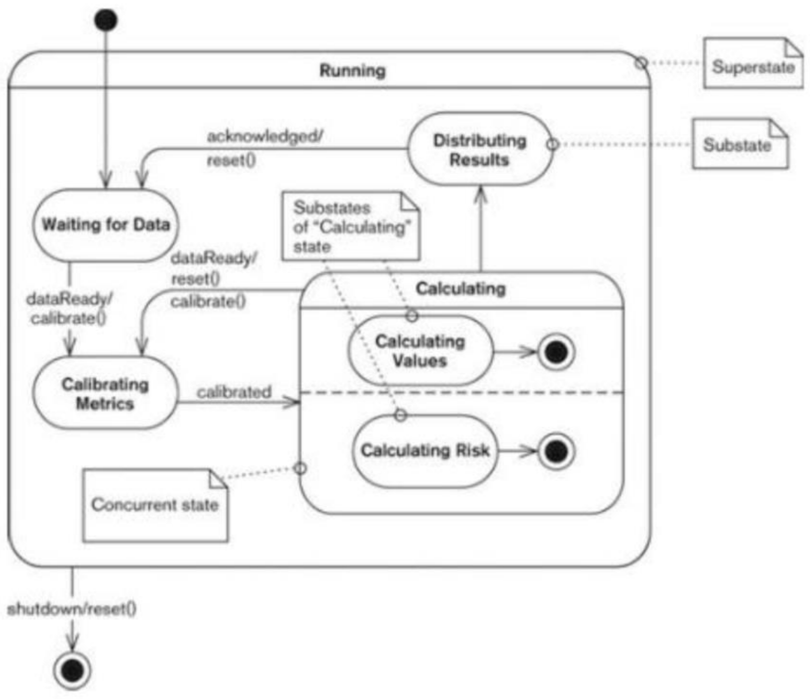
- Volumetric models

4.4. STAKEHOLDERS OF INFORMATION VIEW

- Primary users
- Acquirers
- Developers
- Testers
- Maintainers

5. CONCURENCY VIEWPOINT

5.1. ELEMENTS OF STATE MODEL IN CONCURENCY VIEW



5.2. CONCERNS OF CONCURENCY VIEW

- Task structure
- Inter-process communication
- State management
- Synchronization and integrity (consistency)
- Supporting scalability
- Startup and shutdown
- Task failure
- Reentrancy

5.3. MODELS OF CONCURENCY VIEW

- **System-level concurrency model**
  - Sequential VS Concurrent VS Parallel
  - Threads VS Process
- **State model**
  - Entities of state model:
    - **State** – Named, stable condition during runtime of functional element's lifetime. States can be associated with "waiting for something"
    - **Transition** – allowable change from one state to another, following in an occurrence of event
    - **Event** – indication that something of an interest in the system has happened. Events trigger transitions between element states
    - **Actions** – atomic pieces of processing that can be associated with a transition.
  - Activities of state model:
    - Define notation
    - Identify the states
    - Design state transitions
- **Process model**
  - Map the elements to the tasks
  - Determine threading design

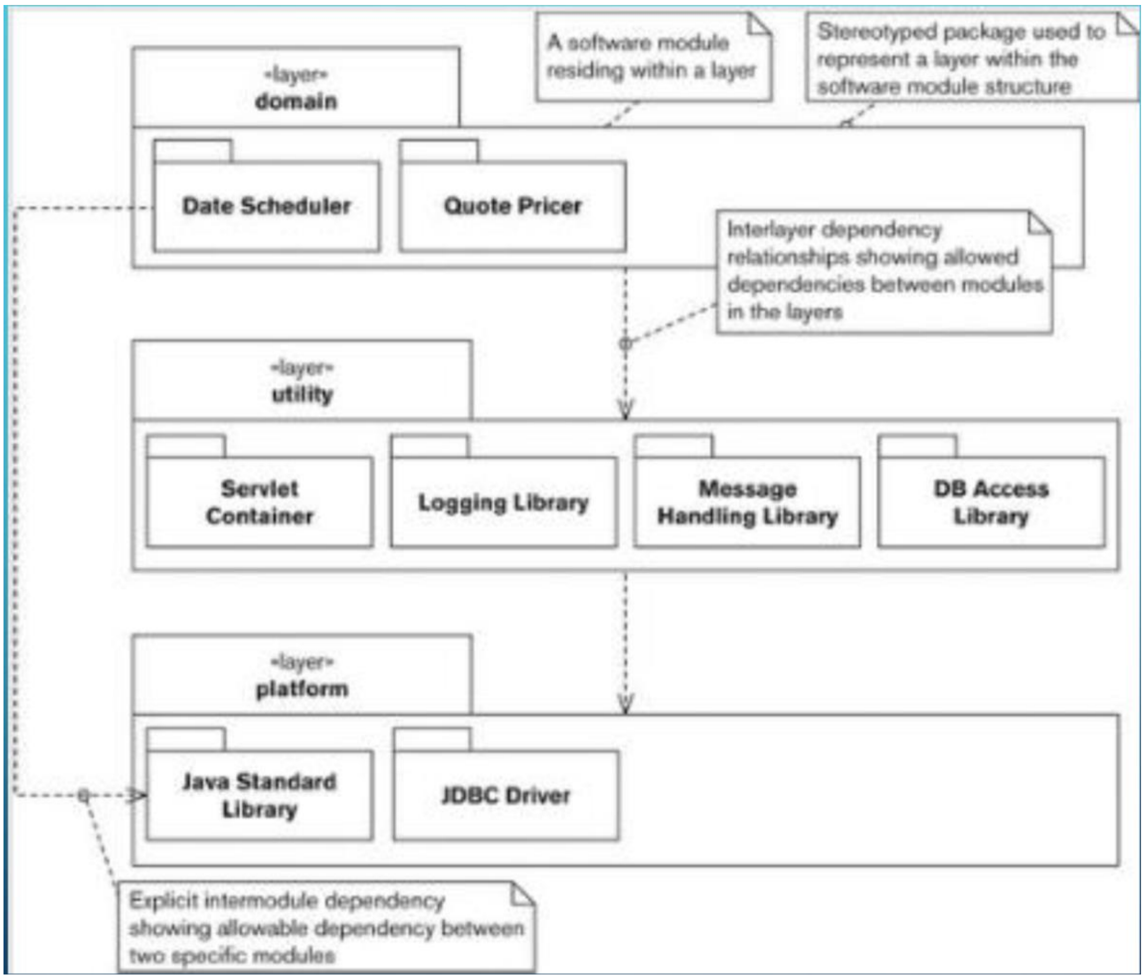
    - Define mechanisms for resource sharing
    - Define IPC mechanisms
    - Assign priorities to threads and processes
    - Analyze deadlocks
    - Analyze contention
- **Actor model** - ASYNCHRONOUS, DISTRIBUTED, MESSAGE BASED MODEL TO SUPPORT CONCURRENT OPERATIONS
    - Actions:
        - Can receive messages
        - Can do computations
        - Can create other actors
        - Can decide what to do when next message arrive
        - Can contain private state
    - Notes:
        - USUALLY ACTORS RUN IN ISOLATED MODE I.E. PROCESS LEVEL, MEANING NO SHARED RESOURCES LIKE MEMORY
        - ACTORS HAVE ADDRESSES OR MAILBOXES TO ALLOW MESSAGE DELIVERY
        - ACTORS CAN QUEUE MESSAGES IN MAILBOXES
        - ACTOR MODEL CAN SCALE WELL ON MULTIPLE MACHINES DUE TO IT'S DESIGN
        - ACTOR MODEL PROMOTES “LET IT CRASH” PHILOSOPHY DUE TO SOPHISTICATED LEVEL OF STATE RESET AND RECOVERY

5.4. STAKEHOLDERS OF CONCURENCY VIEW

• Communicators
• Developers
• Testers
• Administrators

6. DEVELOPMENT VIEWPOINT

6.1. ELEMENTS OF DEVELOPMENT VIEW



«layer» domain
Date Scheduler
Quote Pricer
A software module residing within a layer
Stereotyped package used to represent a layer within the software module structure
Interlayer dependency relationships showing allowed dependencies between modules in the layers
«layer» utility
Servlet Container
Logging Library
Message Handling Library
DB Access Library
«layer» platform
Java Standard Library
JDBC Driver
Explicit intermodule dependency showing allowable dependency between two specific modules

6.2. CONCERNS OF DEVELOPMENT VIEWPOINT

• Module organization
• Common processing
• Standardization of design
• Standardization of testing
• Instrumentation
• Codeline organization

6.3. MODELS IN DEVELOPMENT VIEW

- ✓ Module structure models
  - o **Activities of structure model:**
    - ▪ Identify and classify modules
    - ▪ Identify module dependencies
    - ▪ Identify layering rules (if layers are used)
- ✓ Common design models
- ✓ Codeline models
  - o Code organization
  - o Code grouping into modules
  - o Directory structure
  - o How source will be built and tested
  - o What type of tests and how regularly
  - o How software will be released
  - o How source will be controlled configuration
  - o What automated tools will be used

6.4. STAKEHOLDERS OF DEVELOPMENT VIEW

- Developers
- Production engineers
- Testers

6.5. GOOD PRACTICES OF DEVELOPER OPERATIONS IN DEVELOPMENT VIEW

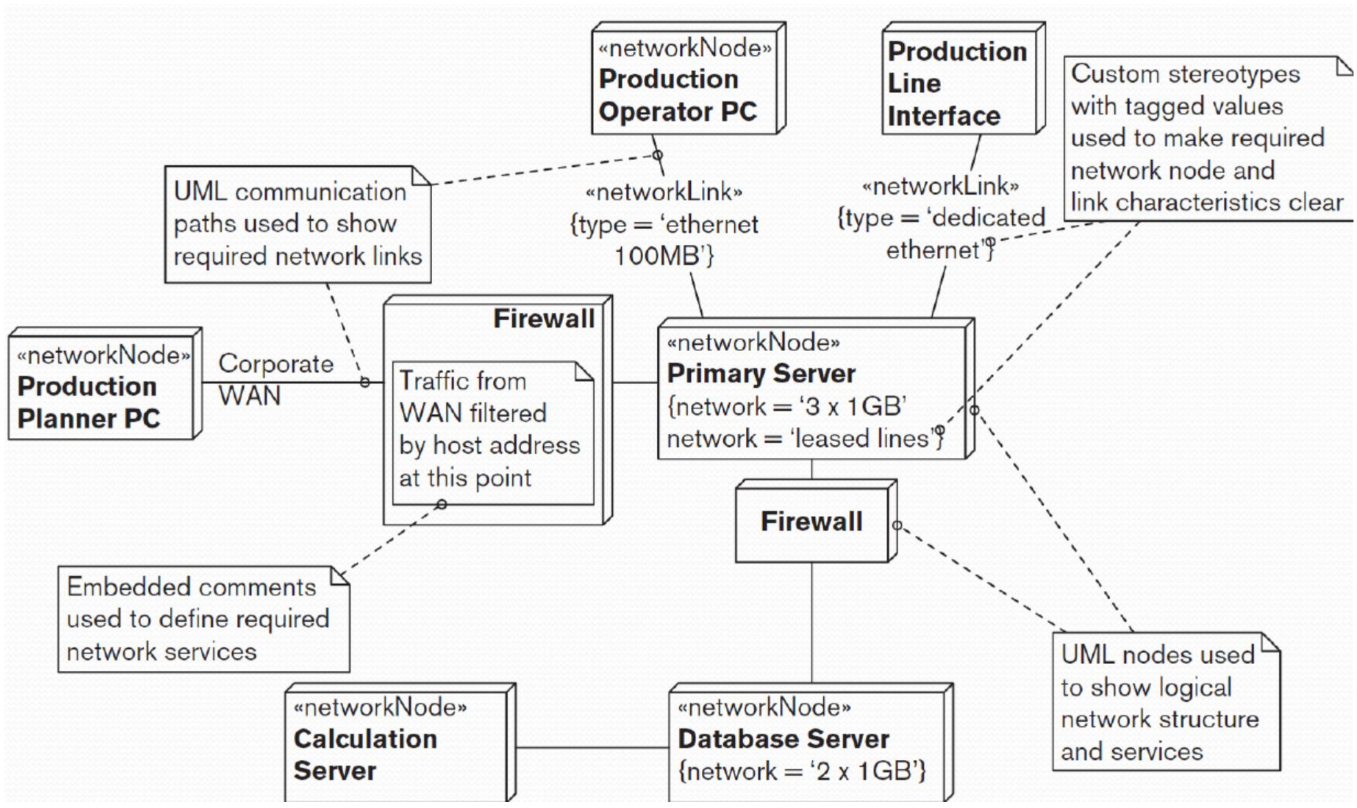
- More than one level of testing
- Automated builds
- Automated deployments
- Feature toggles
- Centralized structured logging
- Centralized qualitative and quantitative monitoring
- Semi-manual rollback support

6.6. GOOD PRACTICES FOR ARCHITECTURAL DECISIONS IN DEVELOPMENT VIEW

- Make view as minimal as possible
- Own developed vs “off the shelf” components
- Maturity and stability assessment for “off the shelf” components
- Interchangeability
- Common design principles to modules and common processing
- Disaster recovery should also be tested, probably in production

7. DEPLOYMENT VIEWPOINT

7.1. ELEMENTS OF DEPLOYMENT VIEW:



«networkNode» Production Operator PC
Production Line Interface
Custom stereotypes with tagged values used to make required network node and link characteristics clear
UML communication paths used to show required network links
«networkLink» {type = 'ethernet 100MB'}
«networkLink» {type = 'dedicated ethernet'}
«networkNode» Production Planner PC
Corporate WAN
Firewall
Traffic from WAN filtered by host address at this point
«networkNode» Primary Server {network = '3 x 1GB' network = 'leased lines'}
Firewall
Embedded comments used to define required network services
UML nodes used to show logical network structure and services
«networkNode» Calculation Server
«networkNode» Database Server {network = '2 x 1GB'}

8. OPERATIONAL VIEWPOINT

8.1. CONCERNS OF OPERATIONAL VIEW

- Installation and upgrade
    - How do we ensure prerequisites?
    - How do we gain required access?
    - Is it fully automated?
    - What needs to be done during upgrade?
    - Is rollback supported?
- Functional migration
    - Big bang
    - Parallel run
    - Staged migration
- Data migration
    - Auto roll forward no rollback
    - Auto roll forward, manual rollback
    - Auto roll forward, auto rollback
- Monitoring and control
    - Routine operations for operational control:
        - Cleanup
        - Transaction resubmission
        - Startup
        - Shutdown
    - Monitoring facilities
        - Software
        - Infrastructure
        - Categorized
        - Criticality
        - Performance counters
- Alerting
- Configuration management
- Support
    - Business support
    - Technical support
    - Support channels
    - Escalation models
    - Technical support types
        - Networks
        - Infrastructure
    - Support hours (24/7 24/5, business hours only and etc., pre-purchased block hours)
    - Support level TIER 0-4 (L0, L1, L2, L3) for SLA (Service Level Agreement)
    - Classes of incidents (criticality and impact to business)
- Backup and restore
    - 3-2-1 backup model (3 copies - 2 in local HDD’s & 1 in the cloud)
    - Disaster recovery
        - Operational handbooks
        - Service/application registry
    - Chaos engineering
        - Build hypothesis
        - Run experiments in production
        - Minimize blast radius
        - Run continuously
- Operation in 3rd party environments

8.2. MODELS OF OPERATIONAL VIEW

- ✓ Installation models

  NOTE: Architectural description should not include installation guides or detailed plans
  - o Installation groups
  - o Dependencies
  - o Constraints
  - o Back-out strategy
- ✓ Migration models
- ✓ Configuration management models
- ✓ Administration models
- ✓ Support models

8.3. STAKEHOLODERS OF OPERATIONAL VIEW

- System administrators
- Production engineers
- Developers
- Testers
- Communicators
- Assessors

8.4. GOOD PRINCIPLES FOR OPERATION VIEW:

- Separate logging and monitoring systems
- Monitor a system as is (no rework/programming required from monitored system perspective)
- Define when and how critical alerts should be sent to responsible people
- Alert not only errors but system or business level information (service started, service stopped, etc.)