CSE 301
Software Design and Architecture

3. Introduction to Software Architecture

Dr. Salisu Garba
Salisu.garba@slu.edu.ng

Outline

- Introduction to software architecture
- Architecture vs Design
- Architect’s responsibilities
- Architecture Business Cycle
- Architectural Representation

Software Architecture

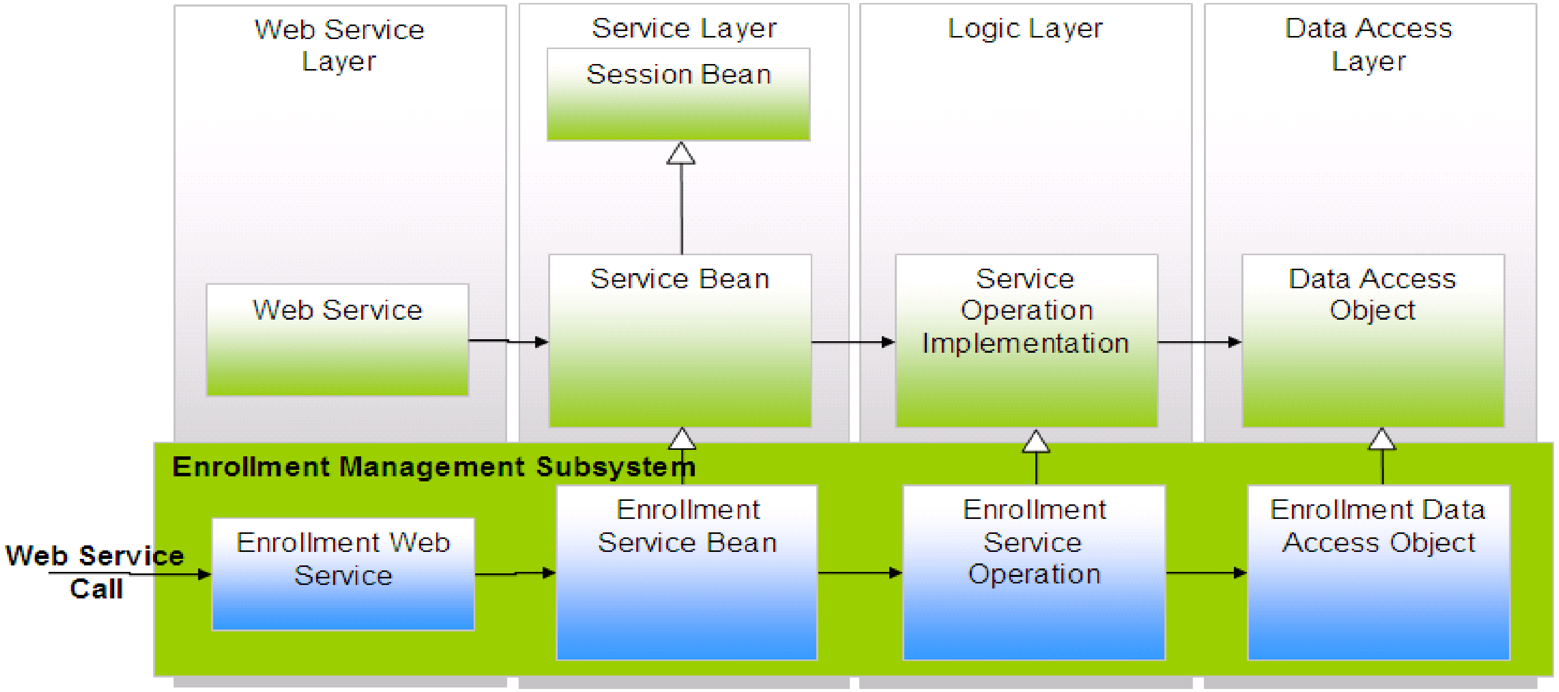
- Software Architecture is the description of **elements** from which system is built, **interactions** among those elements, patterns that guide their composition, and constraints on the patterns.

- Considers system as a **collection of components and their interactions**.

  - **Components** are such things as clients and **servers**, **databases**, **layers**, etc.

  - **Interactions** among components can be procedure calls, shared variable access, etc.

Software Architecture



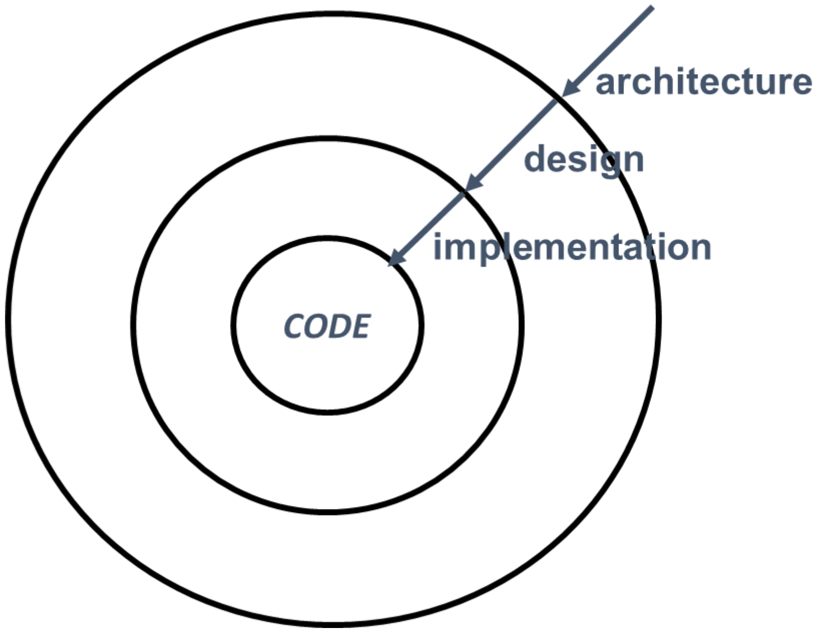
- *What is the nature of the elements?*
- *What are the responsibilities of the elements?*
- *What is the significance of the connections?*
- *What is the significance of the layout?*

Software Architecture

- Where does architecture design come from?
  - The architecture design representation is derived from the **system requirement** specification and the analysis model.
- Who is responsible for developing the architecture design?
  - **Software architects** and designers are involved in this process by translating (mapping) the software system requirements into architecture design.
- Why is software architecture design so important?
  - A poor design may result in a **deficient product** that does not meet system requirements,
- What is the outcome of the software architecture design?
  - A complete software architecture specification must describe not only the elements and connectors between elements, but also the **constraints and runtime behaviors**

Architecture vs. Design

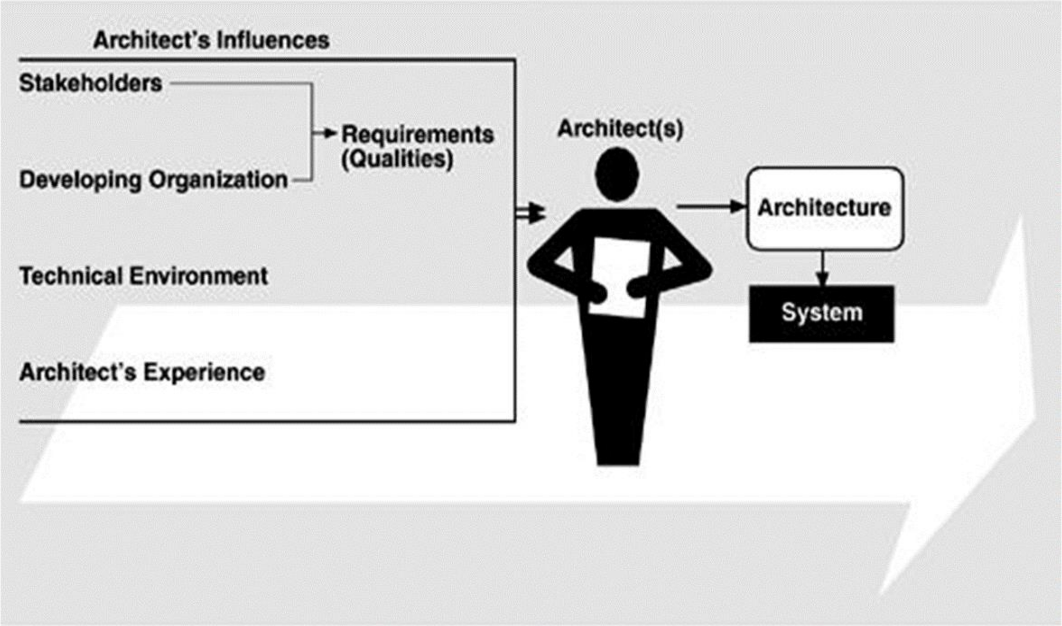
- “**All architecture is design, not all design is architecture**”.
- Architectural design is **outward looking**
  - Focus on **stakeholders**, not technology
- Architecture doesn't describe the complete characteristics of components – Design does.



Architectural influences

- Influences
  - System Stakeholders
  - Developing organization
  - Architects' background and experience
  - Technical environment

- Precautionary measures
  - Know your constraints
  - Early engagement of stakeholders

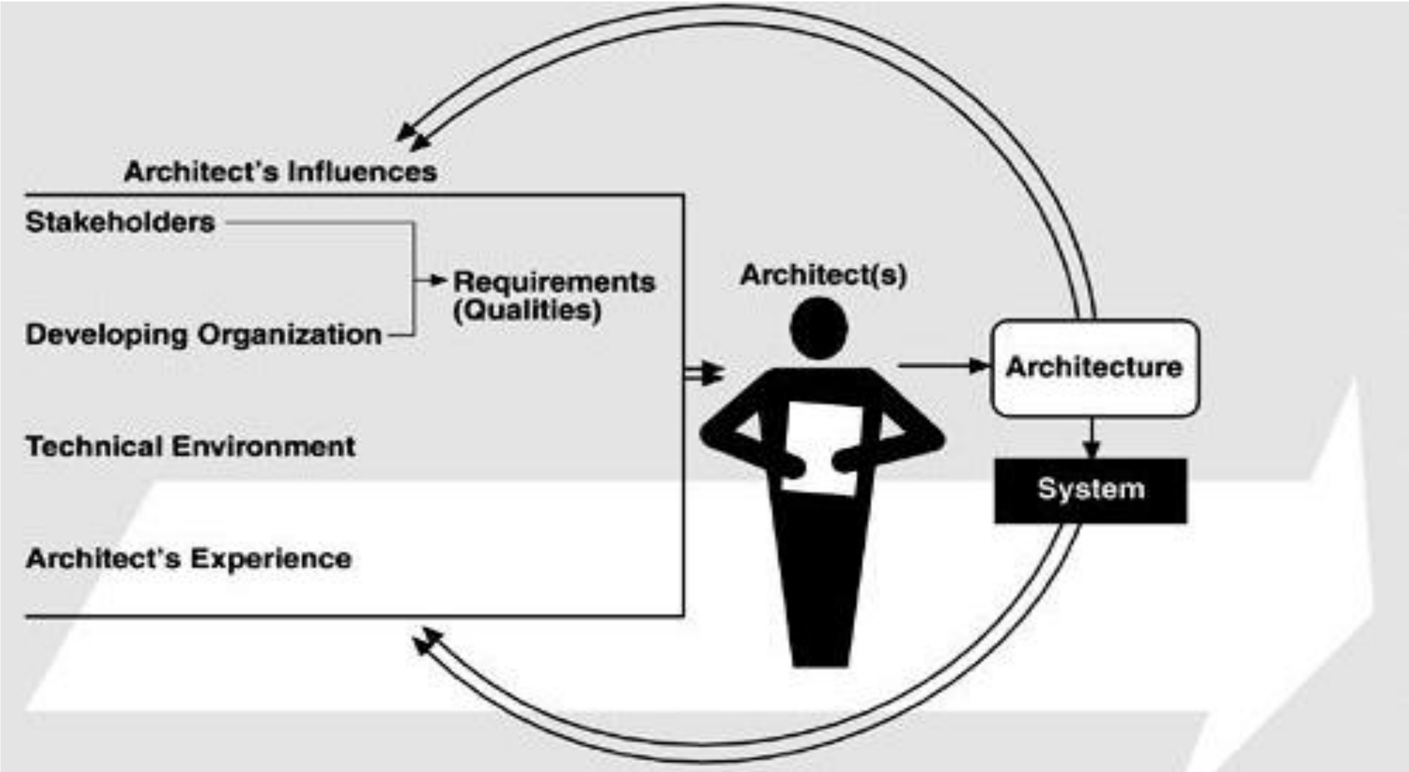


An Architect’s Tasks

- Perform static **partition and decomposition** of a system into subsystems and communications among subsystems.
  - A software element can be configured, delivered, developed, and deployed, and is replaceable in the future.
  - Each element's interface encapsulates details and provides loose coupling with other elements or subsystems.
- Establish dynamic **control relationships among different subsystems** in terms of data flow, control flow orchestration, or message dispatching.
- Consider and evaluate **alternative architecture styles** that suit the problem domain at hand.
- Perform **tradeoff analysis on quality attributes** and other nonfunctional requirements during the selection of architecture styles.
  - For example, in order to increase a distributed system's extensibility, portability, or maintainability, software components and Web services may be the best choice of element types, and a loose connection among these elements may be most appropriate.

Architecture Business Cycle (ABC)

- Software architecture is a result of technical, business and social influences. These are in turn affected by the software architecture itself.
- This cycle of influences from the environment to the architecture and back to the environment is called the *Architecture Business Cycle (ABC)*.



ABC Activities

- **Creating the business case for the system**
  - Why we need a new system, what will be its cost? Time to market, integration with existing systems?
- **Understanding the Requirements**
  - Various approaches for requirements elicitation i.e., object-oriented approach, prototyping. The desired qualities of a system shape the architectural decisions
- **Creating/selecting the architecture**
- **Communicating the architecture**
  - Inform all stakeholders (i.e., developers, testers, managers, etc.)
  - Architecture's documentation should be unambiguous
- **Analysing or evaluating the architecture**
  - Evaluate candidate designs
  - Architecture maps the stakeholders' requirements/needs
- **Implementation based on architecture**
- **Ensuring conformance to an architecture**

Architectural Representations

- Software architecture specifies a **high level of software system** abstraction by employing decomposition, composition, architecture styles, and quality attributes.

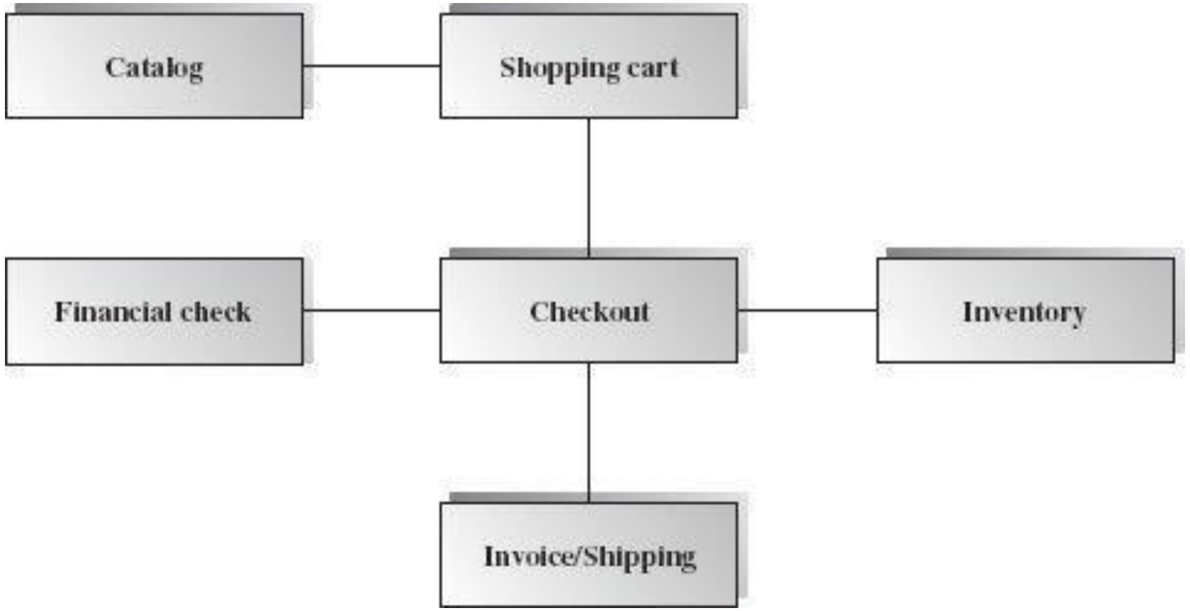
- Every software architecture must describe its collection of components and the connections and interactions among these components.

- It must also specify the deployment configuration of all components and connections.

- Additionally, a software architecture design must conform to the project's functional and nonfunctional requirements.

Architectural Representations – Box diagrams

- **Box-and-line diagrams** are often used to describe the business **concepts and processes** during the analysis phase of the software development lifecycle.
- These diagrams come with descriptions of **components and connectors**, as well as other descriptions that provide common intuitive interpretations.



UML for Software Architecture

- The Unified Modeling Language (UML) is a **graphical language** for
  - visualizing,
  - specifying,
  - constructing, and
  - documenting

the artifacts of a software-intensive system.

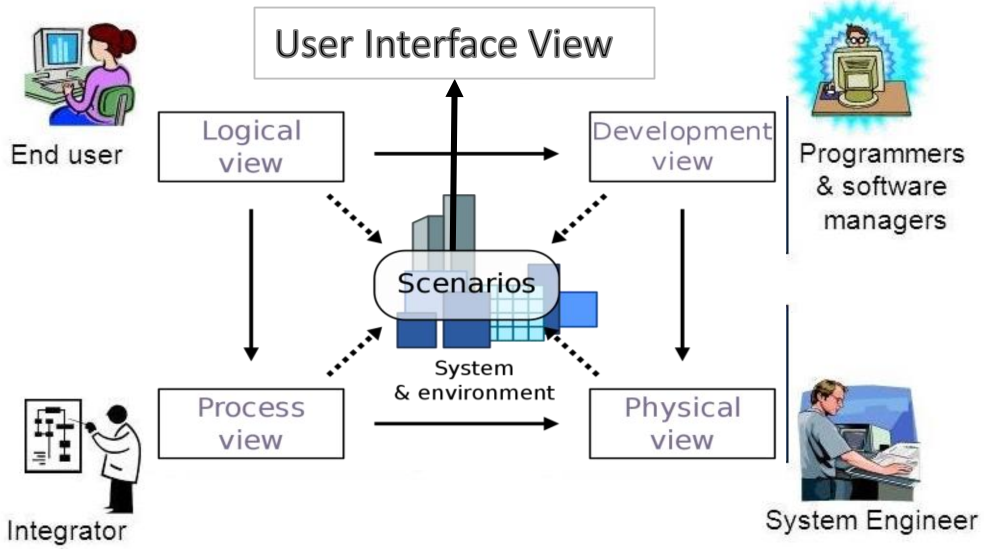
- UML offers a standard way to **draw a system's blueprints**.

Architecture View Models

- A model is a **complete, simplified description** of a system from a particular perspective or **viewpoint**.
- There is **no single view** that can present **all aspects** of complex software to stakeholders!!!
- Software designers can organize the description of their architecture decisions in different views.
- View models provide **partial representations** of the software architecture to specific **stakeholders** such as
  - the system users,
  - the analyst/designer,
  - the developer/programmer,
  - the system integrator, and
  - the system engineer.

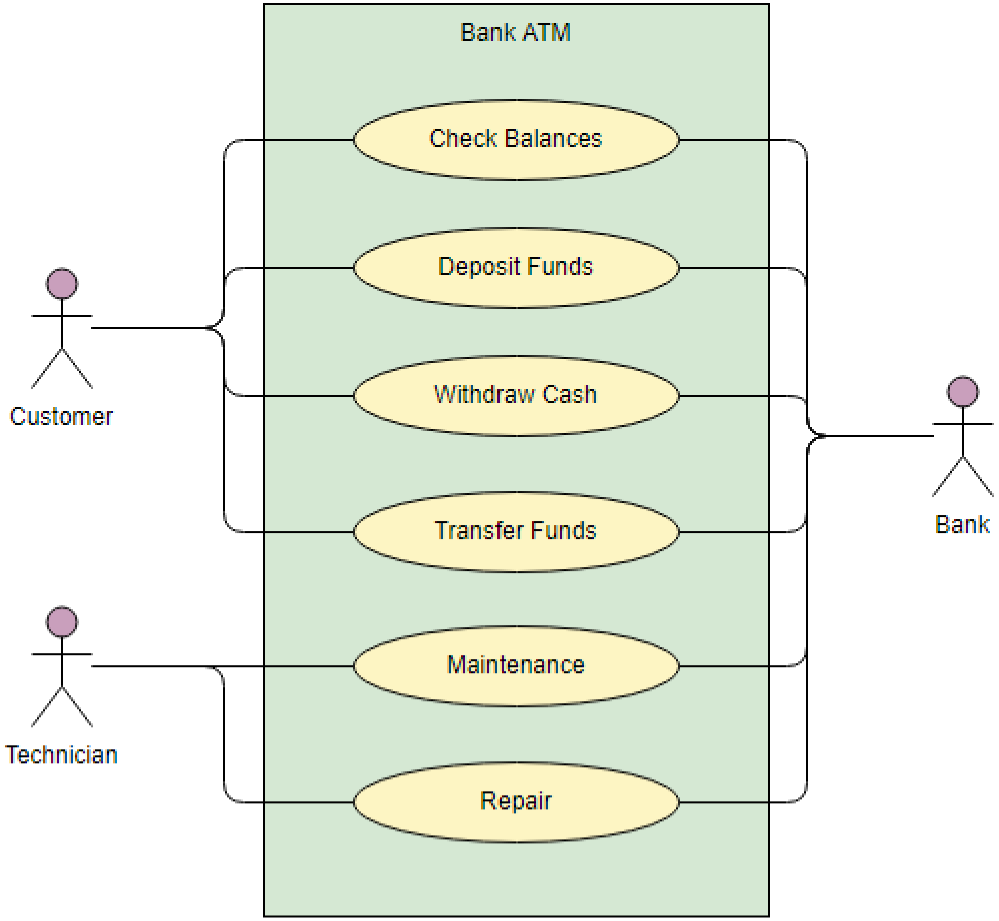
The 4 +1 View Model

- The 4+1 view model was originally introduced by Philippe Kruchten (Kruchten, 1995).
- A multiple-view model that addresses different aspects and concerns of the system.
- A standardizes the software design documents and **makes the design easy to understand by all stakeholders.**
- The model provides four essential views:
  - the logical view,
  - the process view,
  - the physical view, and
  - the development view
- and fifth is **the scenario view**



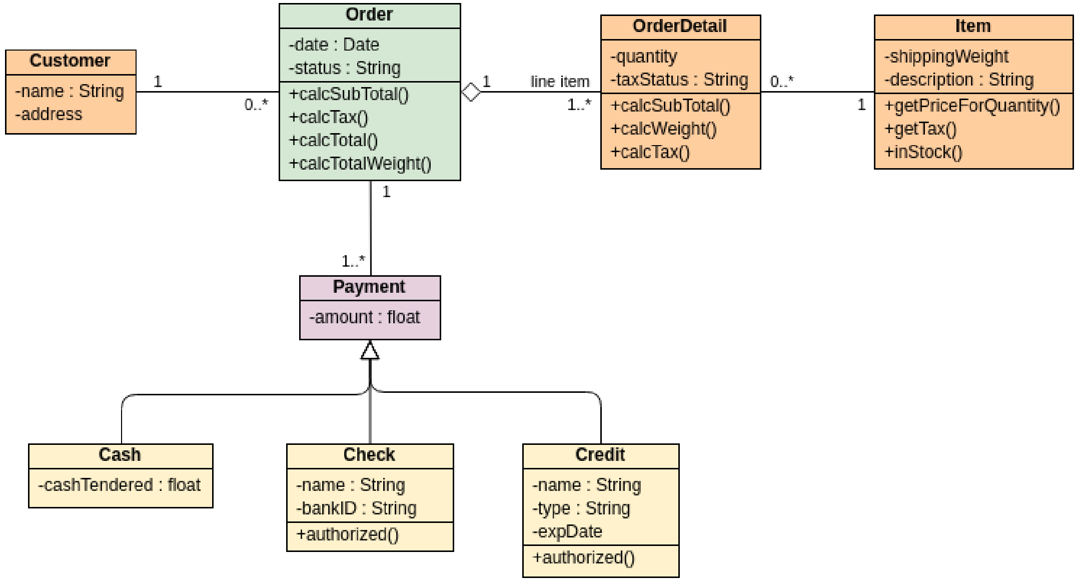
The Scenario View

- The scenario view describes the **functionality** of the system, i.e., how the user employs the system and how the system provides **services to the users**.
- It helps designers to discover architecture elements during the design process and to validate the architecture design afterward.
  - The UML **use case diagram** and other verbal documents



The Logical or Conceptual View

- The logical view is based on application domain entities necessary to implement the functional requirements.
- The logical view specifies system decomposition into **conceptual** entities (such as **objects**) and connections between them (such as **associations**).
- The logical view is typically supported by
  - UML static diagrams including **class diagrams** and
  - UML dynamic diagrams such as the interaction overall diagram, sequence diagram, communication diagram, state diagram, and activity diagram.



Customer
-name : String
-address

1
0..*

Order
-date : Date
-status : String
+calcSubTotal()
+calcTax()
+calcTotal()
+calcTotalWeight()

1
line item
1..*

OrderDetail
-quantity
-taxStatus : String
+calcSubTotal()
+calcWeight()
+calcTax()

0..*
1

Item
-shippingWeight
-description : String
+getPriceForQuantity()
+getTax()
+inStock()

1
1..*

Payment
-amount : float

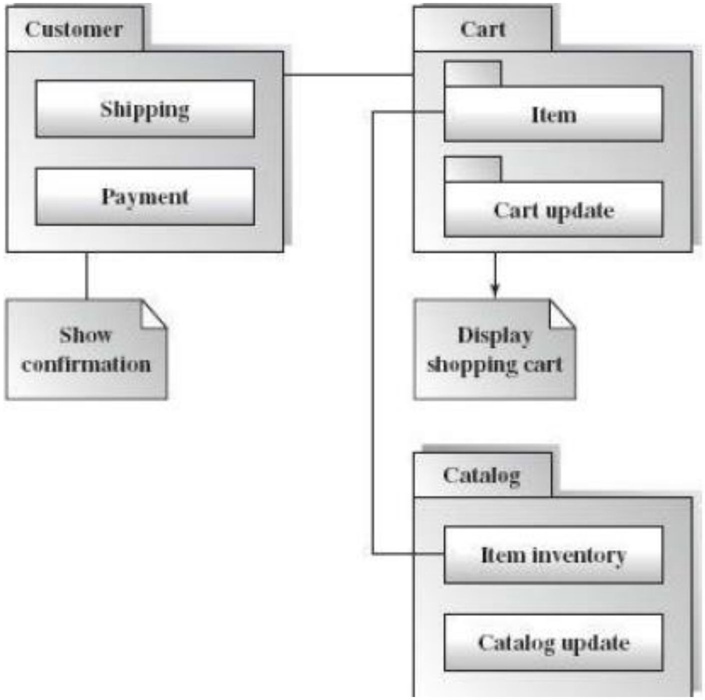
Cash
-cashTendered : float

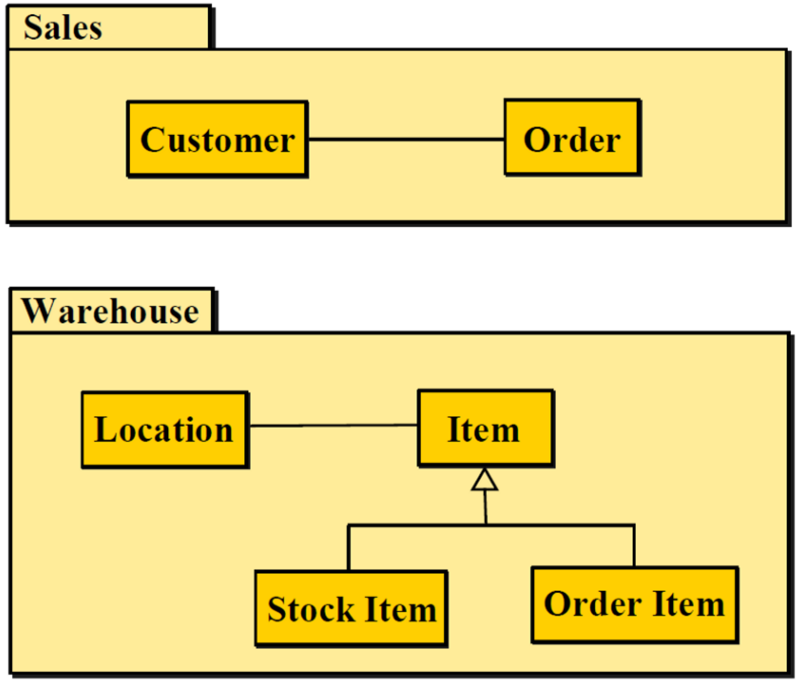
Check
-name : String
-bankID : String
+authorized()

Credit
-name : String
-type : String
-expDate
+authorized()

The Development or Module View

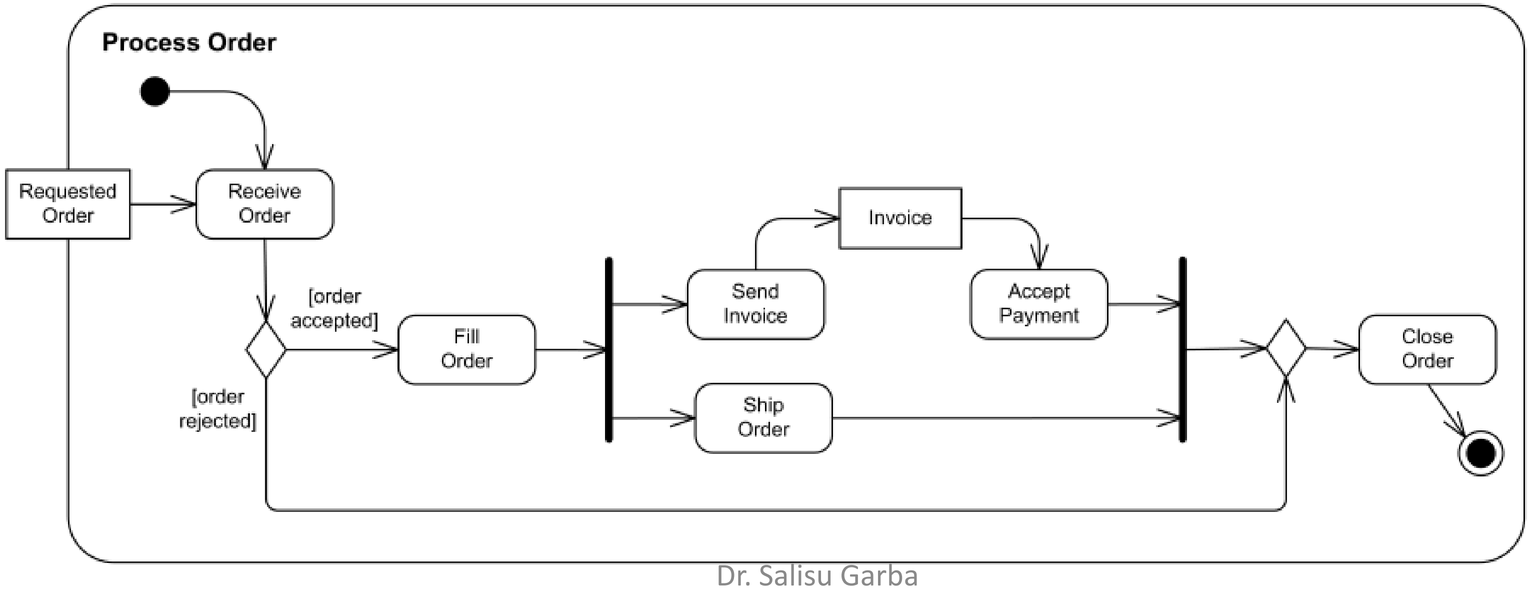
- The development view derives from the logical view and describes the **static organization of the system modules**.
- **Modules such as namespaces, class library, subsystem, or packages** are building **blocks** that group classes for further development and implementation.
- UML diagrams such as **package diagrams** and component diagrams are often used to support this view.





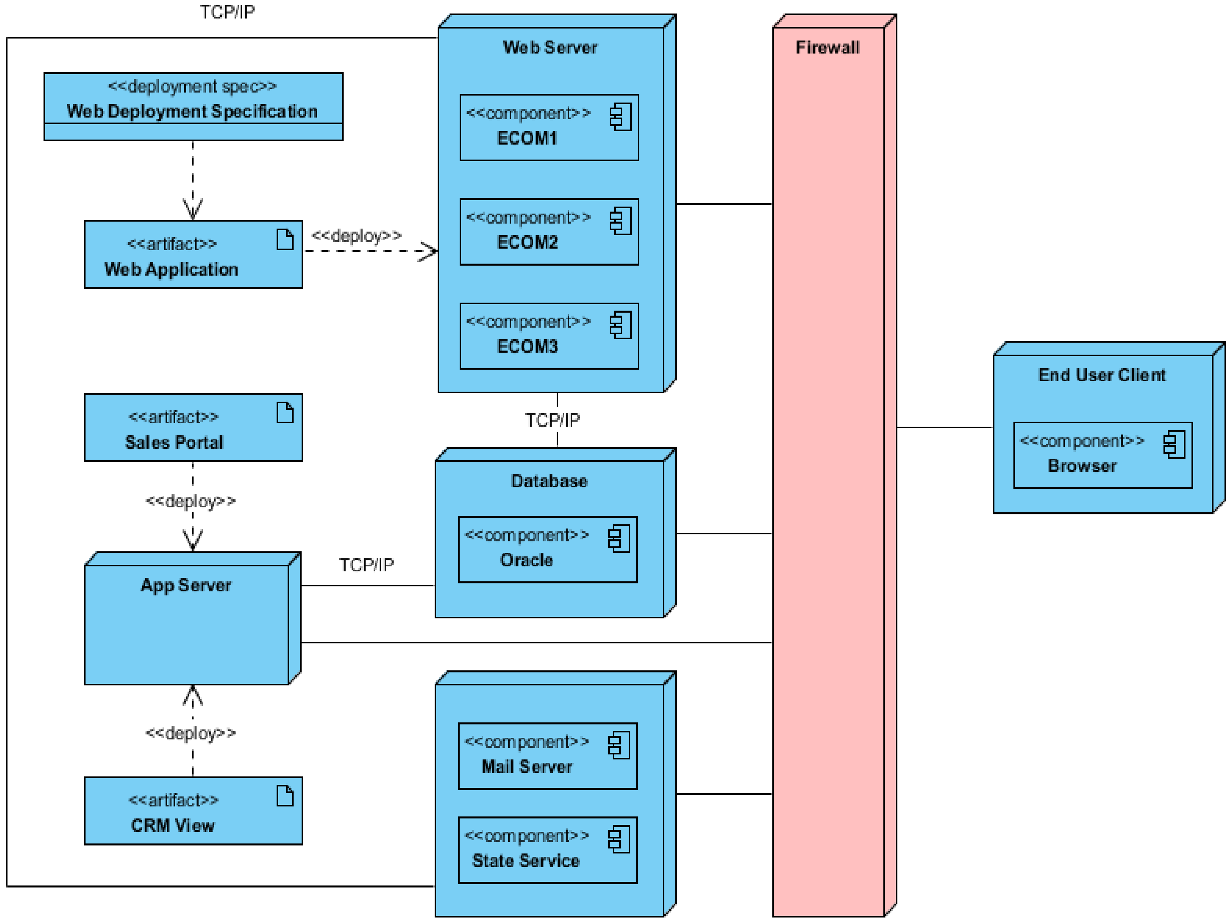
The Process View

- The process view focuses on the **dynamic aspects of the system**, i.e., its **execution time behavior**.
- This view maps functions, activities, and interactions onto runtime implementation.
- The process view takes care of the concurrency and synchronization issues between subsystems.
- The UML **activity diagram** and interaction overview diagram support this view.



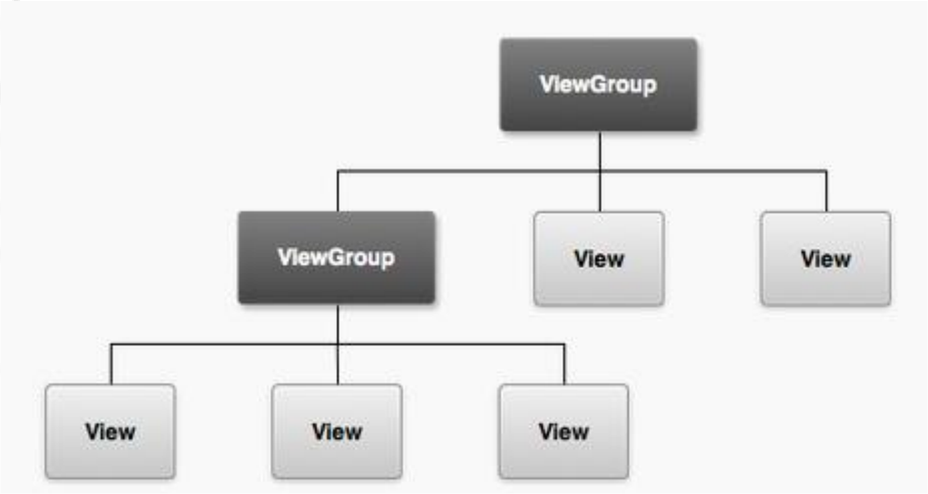
The Physical View

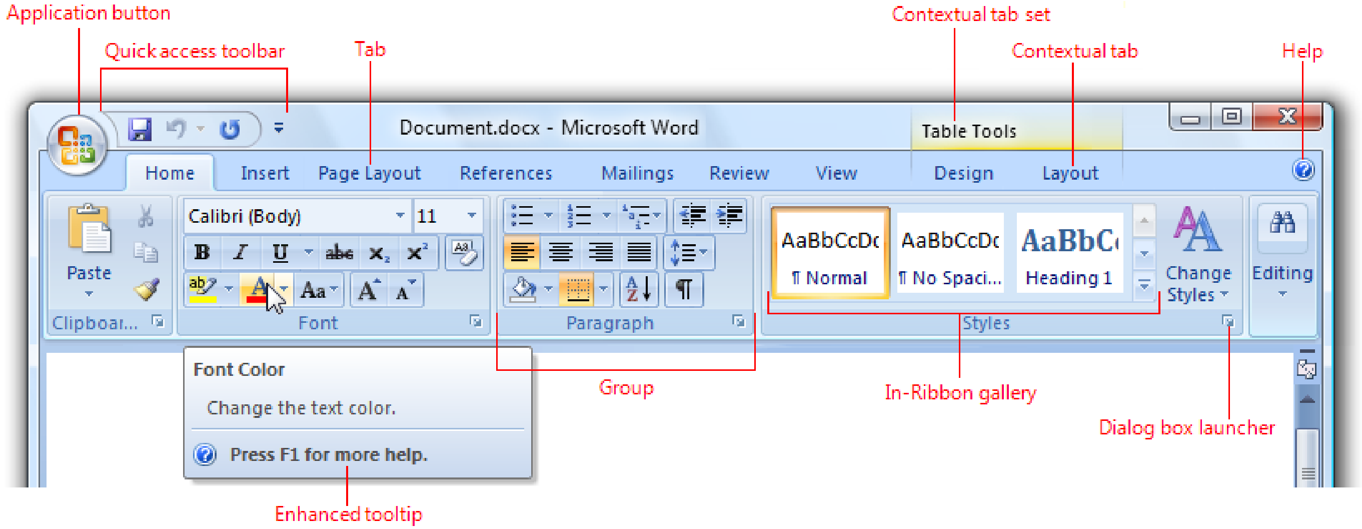
- The physical view describes **installation, configuration, and deployment** of the software application.
- It concerns itself with how to deliver the **deploy-able system**.
- The physical view shows the mapping of software onto hardware.
  - It is particularly of interest in distributed or parallel systems.
- The UML **deployment diagrams** and other documentation are often used to support this view.



The User Interface View

- The User Interface (UI) view is an extended view that provides a **clear user-computer interface** view and **hides implementation details**.
- This view may be provided as a **series of screen snapshots** or a dynamic, interactive prototype demo.
- Any modification on this view will have direct impact on the scenarios view.





Font Color

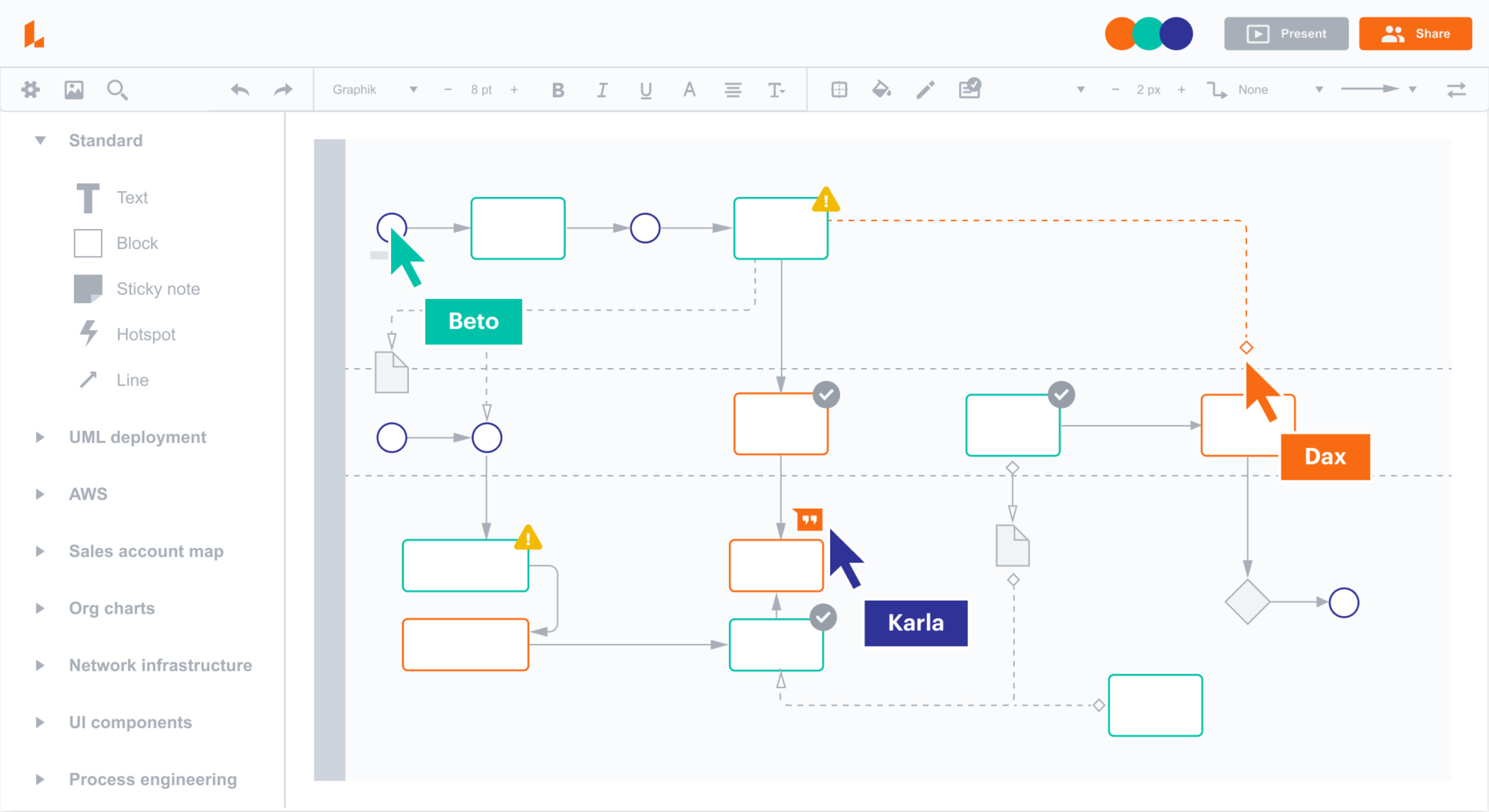
Change the text color.

Press F1 for more help.

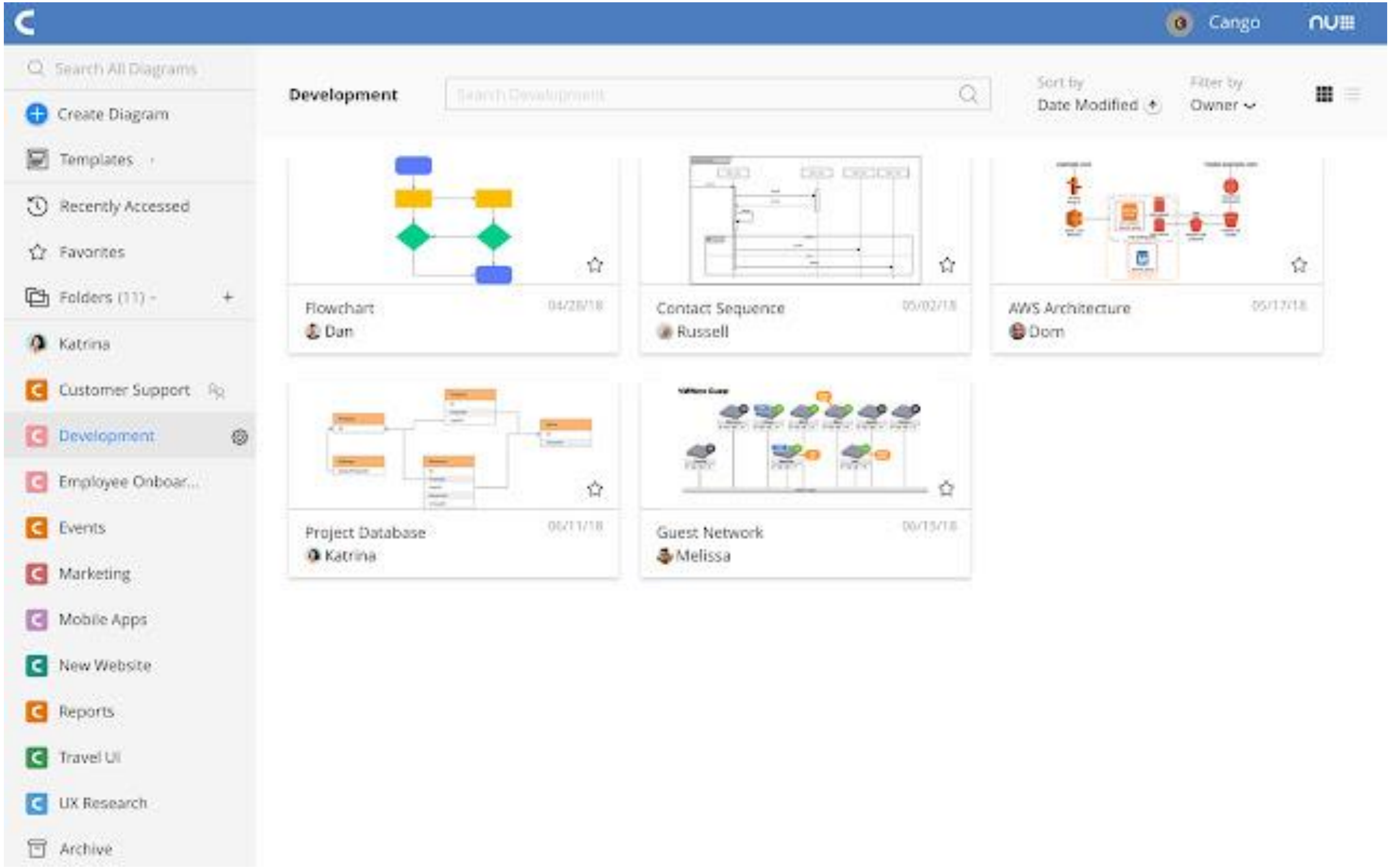
Architecture Description Language

- It is a **modelling notation to support architecture-based development**
- used to define and model system architecture prior to detailed design and implementation
- Architecture Description Languages (ADLs) can be used to specify an architecture
  - UML (OMG) - general-purpose
  - SADL (SRI)
  - Aesop (Carnegie Mellon University)
  - Acme (CMU) – and interchange format
  - Rapide (Stanford University)
  - Wright (CMU)
  - Darwin (Imperial College London)

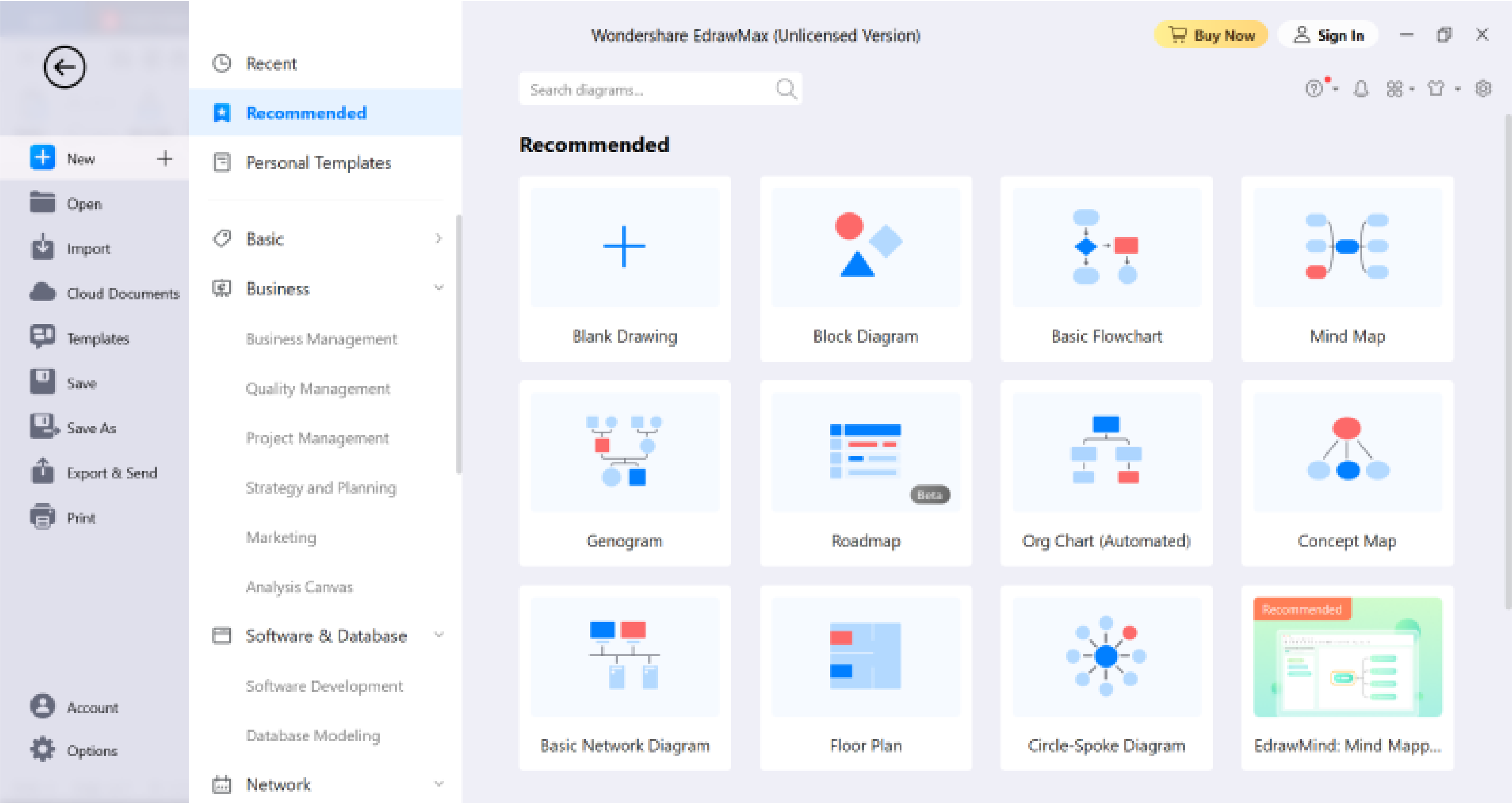
Tools - Lucidchart



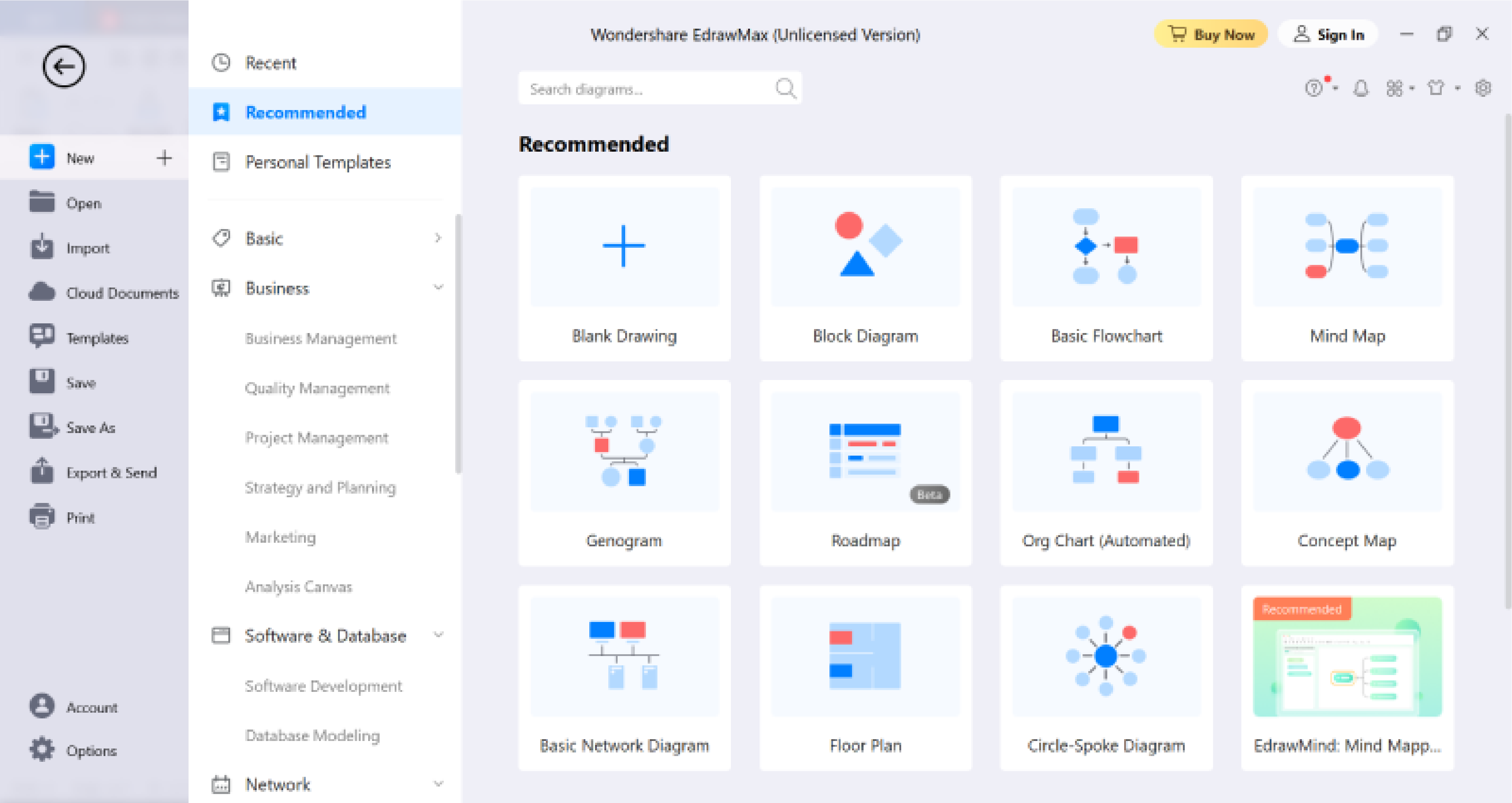
Tools - Cacoo


Cango
Search All Diagrams
Create Diagram
Templates
Recently Accessed
Favorites
Folders (11)
Katrina
Customer Support
Development
Employee Onboar...
Events
Marketing
Mobile Apps
New Website
Reports
Travel UI
UX Research
Archive
Development
Search Development
Sort by
Date Modified
Filter by
Owner
Flowchart
04/28/18
Dan
Contact Sequence
05/02/18
Russell
AWS Architecture
05/17/18
Dom
Project Database
06/11/18
Katrina
Guest Network
06/15/18
Melissa

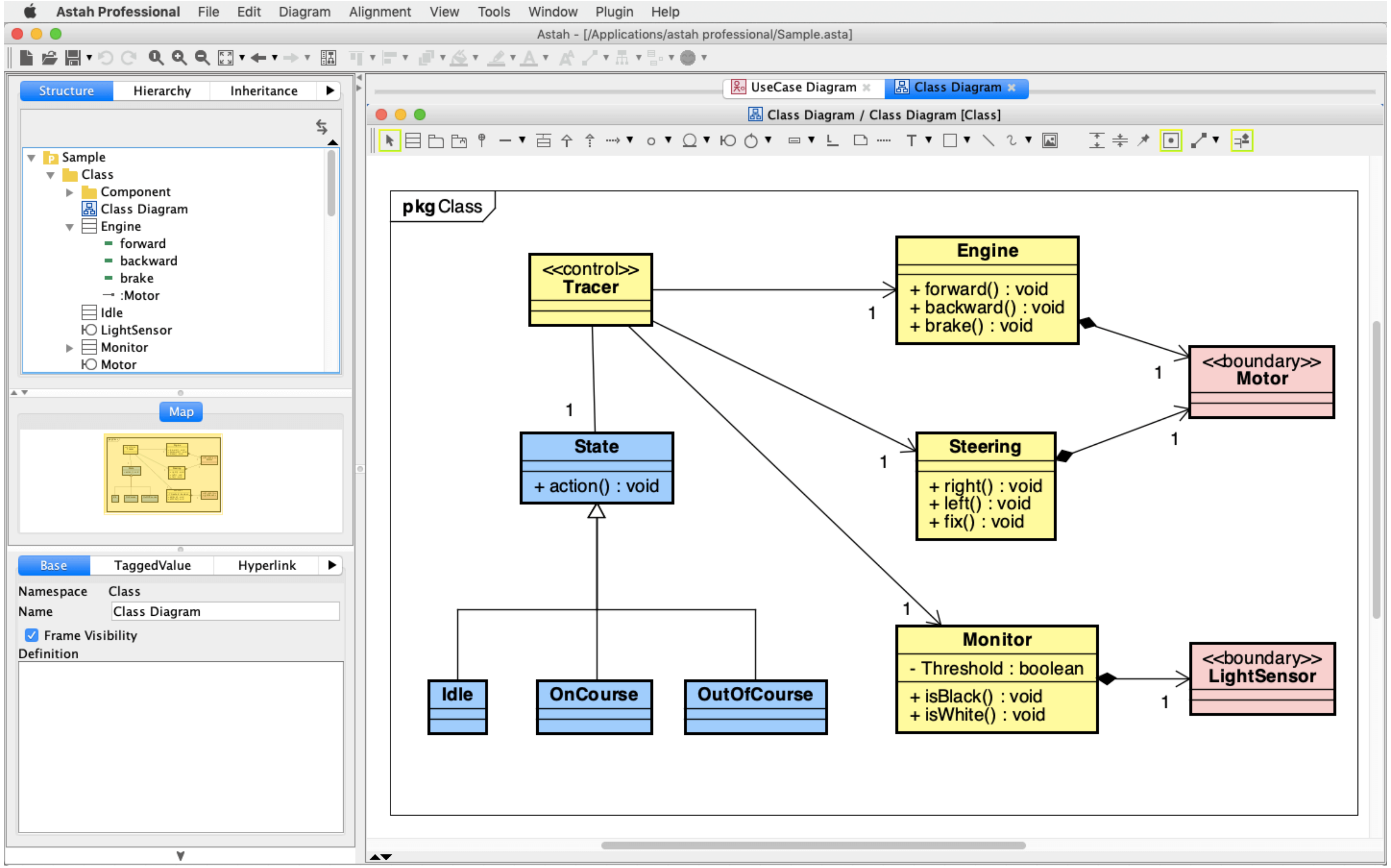
Tools - EdrawMax



Tools - Microsoft Visio Pro



Tools - Astah



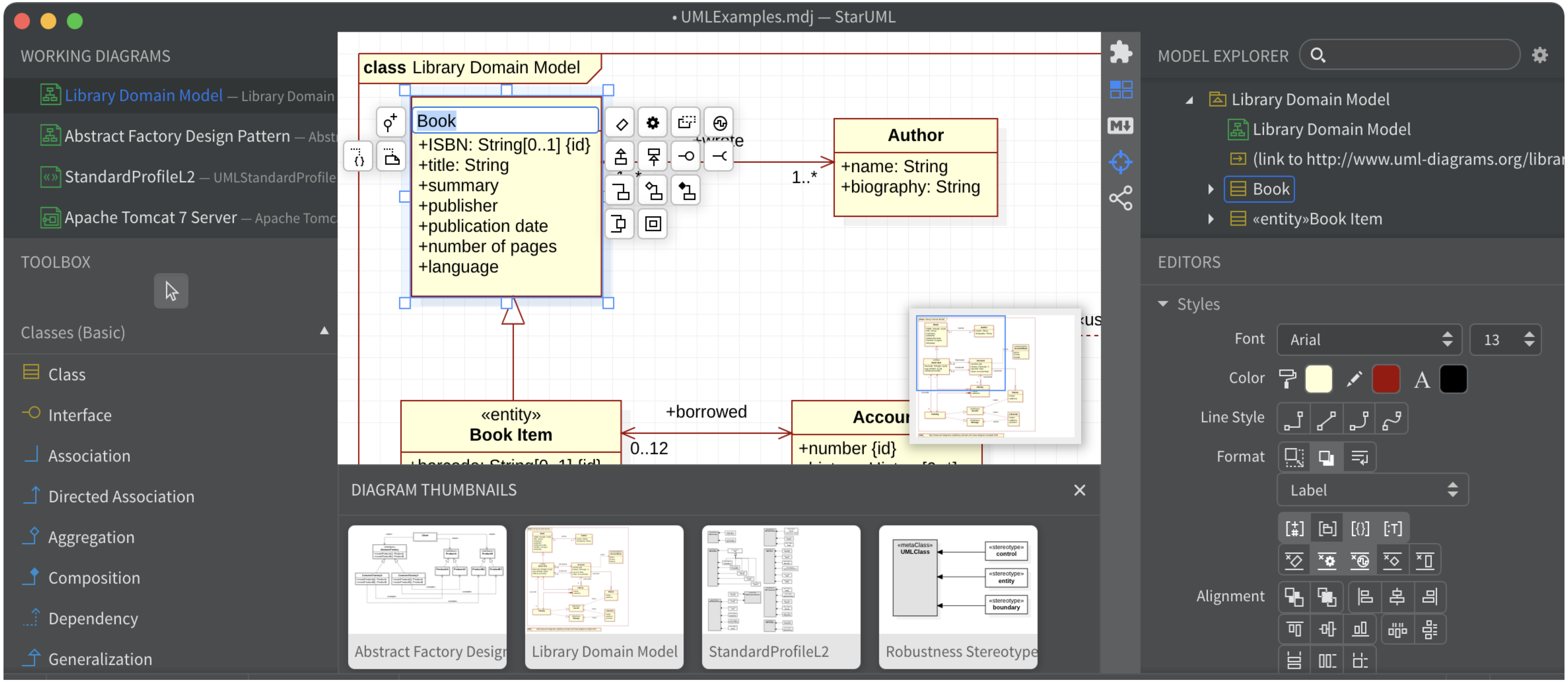
Namespace Class

Name Class Diagram

☑ Frame Visibility

Definition

Tools - StarUML



Summary

- Introduction to software architecture
- Architecture vs Design
- Architect’s responsibilities
- Architecture Business Cycle
- Architectural Representation

Q & A

Any Question?