Operation Contract

- Use cases are the primary mechanism in the unified process to describe the system behavior and use are usually sufficient.
- However, sometimes more detailed description of system behavior has value.
- Contract describe the detail system behavior in term of state to change to object in the domain model.

Operation Contract

- Contract may be defined for the system operation i.e operation that the system as a black box offer in its public interface to handle incoming system events. System operation can be identified by discovering these system events.
- Operation contract describe how the internal state of the concepts in the domain model may change.
- Operation contracts are described in terms of preconditions and post conditions.
- Operation contracts are define in terms of system operations.
- System operation handle system events.

Writing Operation Contract

- **Operation –** Name of the Contract (Parameter).
- **Cross Reference –** The use cases in which the OC occurs.
- **Preconditions –** Noteworthy assumptions about state of system or objects in DM before execution.
- **Postconditions –** State of Objects in DM after execution of operation.

Examples of Operation Contract

Contract CO1: makeNewSale **Operation:**

makeNewSale() **References:** Use cases: Process Sale **Pre-condition:** none

Post-condition: A sale instance "s" was created (instance creation). S was associated with the Register (association form). Attribute of "s" were initialized.

Examples of Operation Contract

Contract CO2: enterItem

Operation: enterItem(itemId: itemId, quantitiy: integer)

References: Use cases: Process Sale

Pre-condition: there is a sale underway

condition: A SalesLineItem instance sli was created (instance creation) sli was associated with the current sale (association formed) sli.quantity became quantity (association modification)

sli was associated with a ProductSpecification based on itemId matched (association formed)

Examples of Operation Contract

Contract CO3: endSale

Operation: endSale()

References: Use cases: Process Sale

Pre-condition: there is a sale underway

Post-condition: Sale.iscomplete() became true (attribute modification)

Guidelines of Making Contract

- Identify the sequence diagram from the system sequence diagram.
- For system operation that are complex and perhaps subtle in their results or which are not clear in the use case, construct a contract.
- To describe the post condition use the following category.
  - Instance creation and deletion.
  - Attribute Modification
  - Association formed and broken

Concept

- A state machine is any device that stores the status of an object at a given time and can change status or cause other actions based on the input it receives. States refer to the different combinations of information that an object can hold, not how the object behaves.
- They are especially important in modeling the behavior of an interface, class, or collaboration.
- State diagrams emphasize the event-ordered behavior of an object, which is especially useful in modeling reactive systems.

Key Concept of State Machine

- A **state machine** is a behavior that specifies the sequences of states an object goes through during its lifetime in response to events, together with its responses to those events.
- A **state** is a condition or situation during the life of an object during which it satisfies some condition, performs some activity, or waits for some event.
- An **event** is the specification of a significant occurrence that has a location in time and space. In the context of state machines, an event is an occurrence of a stimulus that can trigger a state transition.

Key Concept of State Machine

- A **guard condition** is evaluated after the trigger event for the transition occurs. It is possible to have multiple transitions from the same source state and with the same event trigger, as long as the guard conditions don't overlap.
- A **transition** is a relationship between two states indicating that an object in the first state will perform certain actions and enter the second state when a specified event occurs and specified conditions are satisfied. Activity is an ongoing non-atomic execution within a state machine.
- An **action** is an executable atomic computation that results in a change in the state of the model or the return of a value.

Application of State Machine Diagram

- Illustrating use case scenario in a business context.
- Describing how an object moves through various state within its lifetime.
- Showing overall behavior of the state or the behavior of the related set of state machine.

Deployment Diagram

- Deployment diagrams are used to visualize the topology of the physical components of a system, where the software components are deployed.
- Deployment diagrams are used to describe the static deployment view of a system.
- Deployment diagrams consist of nodes and their relationships.

Deployment Diagram

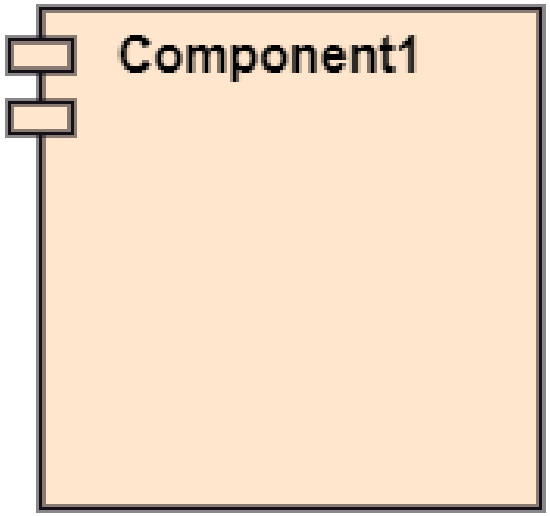
- **Deployment diagram elements**
- A variety of shapes make up deployment diagrams. This list offers an overview of the basic elements you may encounter, and you can see most of these items illustrated in the image below.
- **Artifact:** A product developed by the software, symbolized by a rectangle with the name and the word "artifact" enclosed by double arrows.
- **Association**: A line that indicates a message or other type of communication between nodes.
- **Component:** A rectangle with two tabs that indicates a software element.

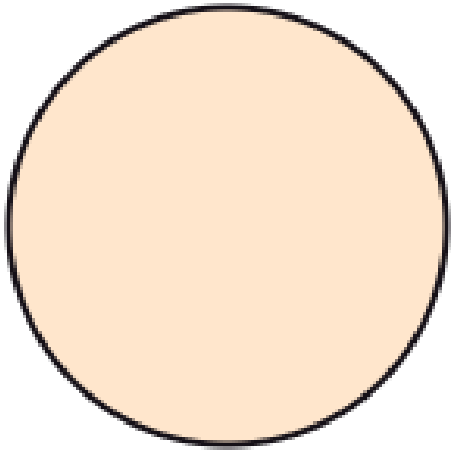
Deployment Diagram

- **Deployment diagram elements**
- **Dependency**: A dashed line that ends in an arrow, which indicates that one node or component is dependent on another.
- **Interface:** A circle that indicates a contractual relationship. Those objects that realize the interface must complete some sort of obligation.
- **Node:** A hardware or software object, shown by a three-dimensional box.
- **Node as container:** A node that contains another node inside of it—such as in the example below, where the nodes contain components.
- **Stereotype:** A device contained within the node, presented at the top of the node, with the name bracketed by double arrows.

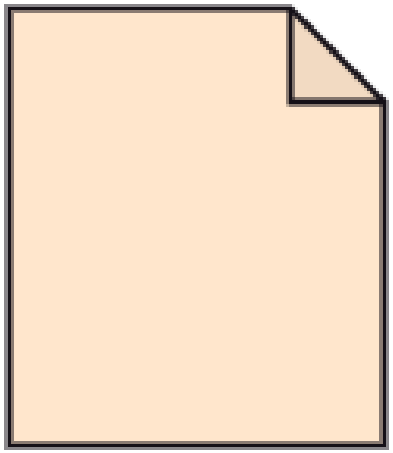
Deployment Diagram

Symbols used in Deployment Diagram

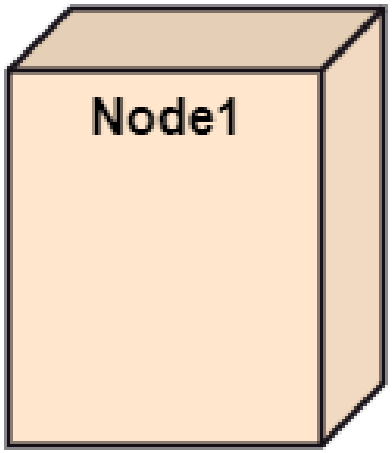




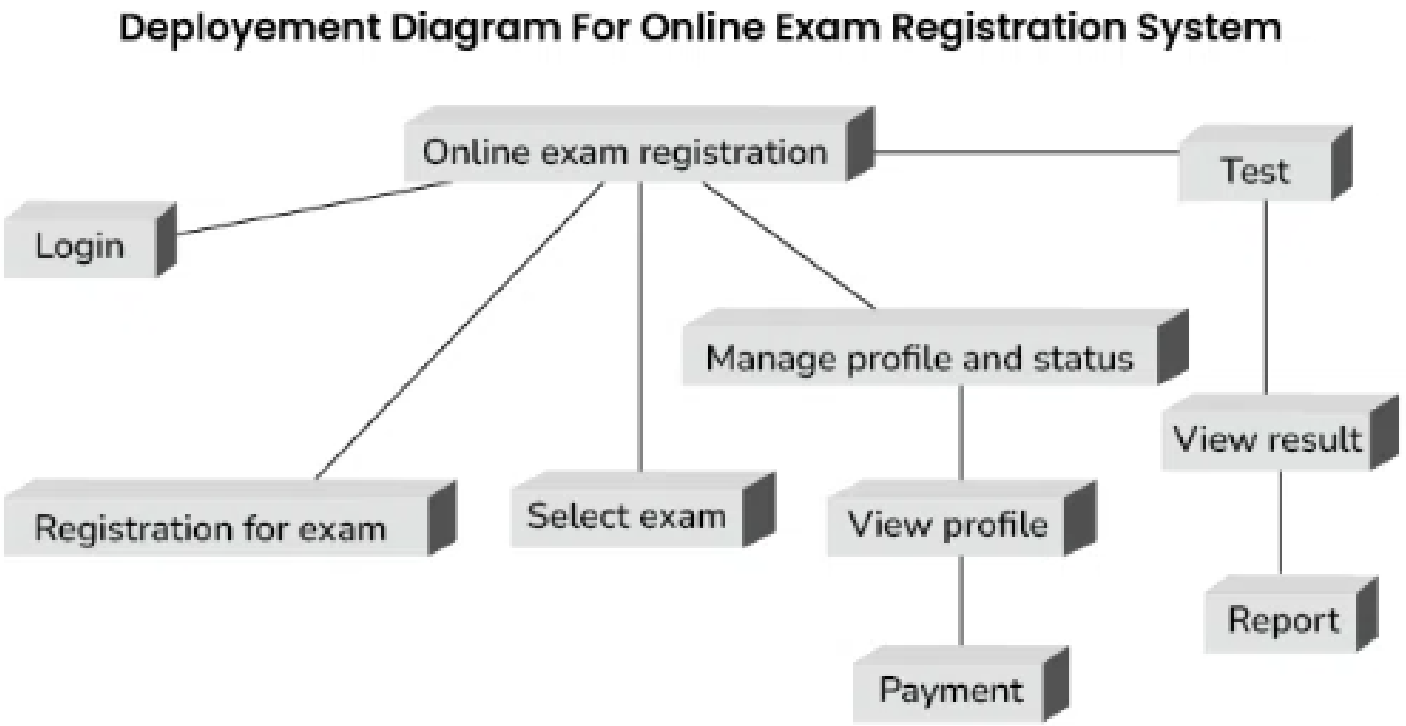
Interface1



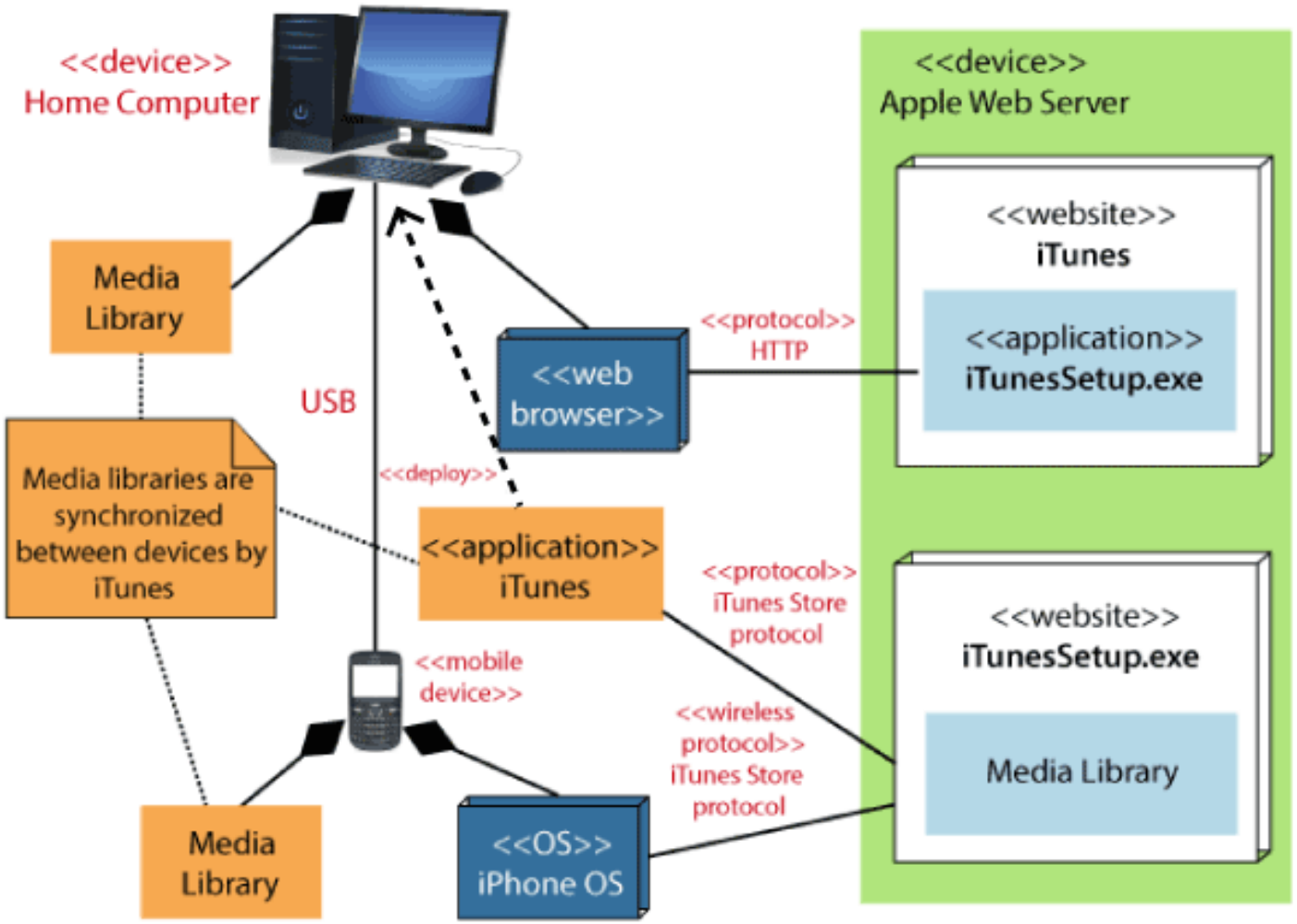
Artifact1



Example of Deployment Diagram



Example of Deployment Diagram



Benefits of Deployment Diagram

- Visual Representation
- Communication Tool
- Planning and Management Aid
- Documentation Support
- Analysis and Optimization
- Testing and Validation Planning

Component Diagram

- A component diagram is used to break down a large object-oriented system into the smaller components, so as to make them more manageable.
- It models the physical view of a system such as executable, files, libraries, etc. that resides within the node.
- It visualizes the relationships as well as the organization between the components present in the system.

Component Diagram

- It helps in forming an executable system. A component is a single unit of the system, which is replaceable and executable.
- The implementation details of a component are hidden, and it necessitates an interface to execute a function.
- It is like a black box whose behavior is explained by the provided and required interfaces.

Component Diagram

- The main purpose of the component diagram are enlisted below:
  - It envisions each component of a system.
  - It constructs the executable by incorporating forward and reverse engineering.
  - It depicts the relationships and organization of components.

Component Diagram