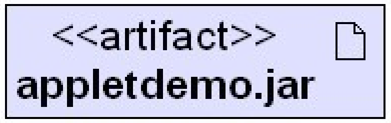
UML Deployment Diagrams

Introduction

- The deployment diagram presents the hardware components and the software elements installed on that environment.
- The deployment diagram maps between the various software components (artifacts) and the hardware units or the software environments (nodes) that will run them.

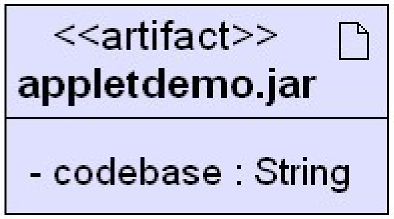
The Artifacts

- The artifacts represent pieces of information related to the software we develop (e.g. DLL file, java byte code file, user guides etc.).
- The notation for an artifact includes an empty rectangle with the artifact name written within and with a small paper, that its top right corner is folded, in the upper right.



Artifact Properties & Operations

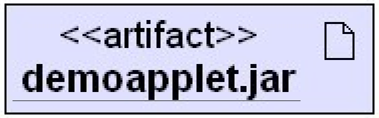
- We can add properties & operations to an artifact in order to represent a set of configurable options (e.g. possible deployment configuration settings).



- The <<artifact>> is not mandatory. This stereotype can be omitted

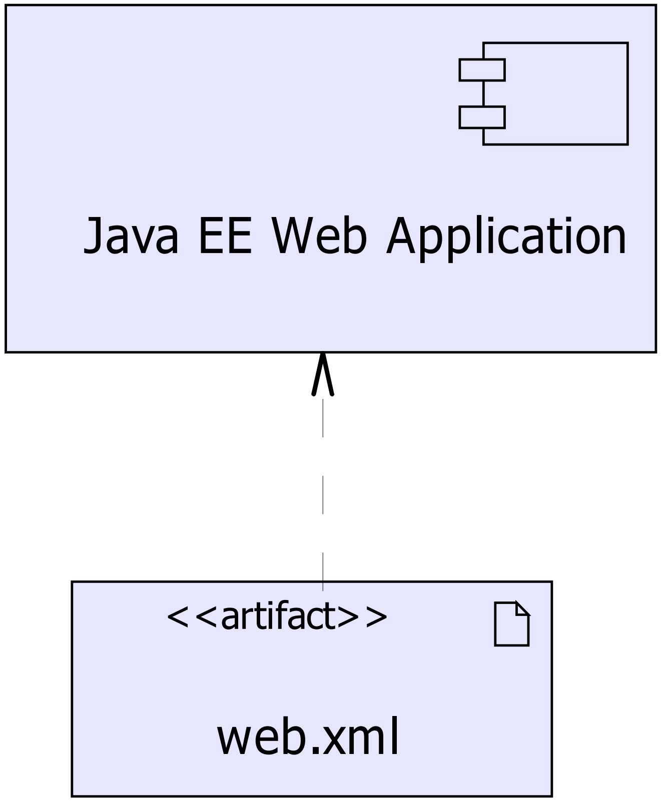
Artifact Instances

- The artifact is a type. As with classes, there is a need to instantiate the artifact.
- An artifact instance is depicted by placing an underline beneath the artifact name. Nevertheless, UML specification allows treating an artifact as an instance and avoid the underlining.



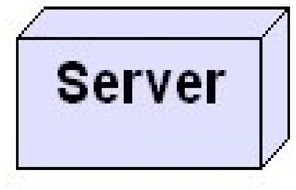
Artifacts & Other UML Elements

- In some cases, the artifact represents another UML element. When that happens we can add a dashed line from the artifact to the element it represents. We can also write `<<manifest>>` instead of writing `<<artifact>>`.



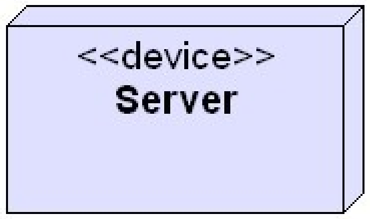
The Nodes

- A node represents a software execution environment or a physical entity (e.g. hardware component). In both cases, the node represents something that can execute an artifact
- The standard notation for a node is a 3D box with the node name written inside. Special icons that represent special hardware components are available as well (e.g. DataBase icon).



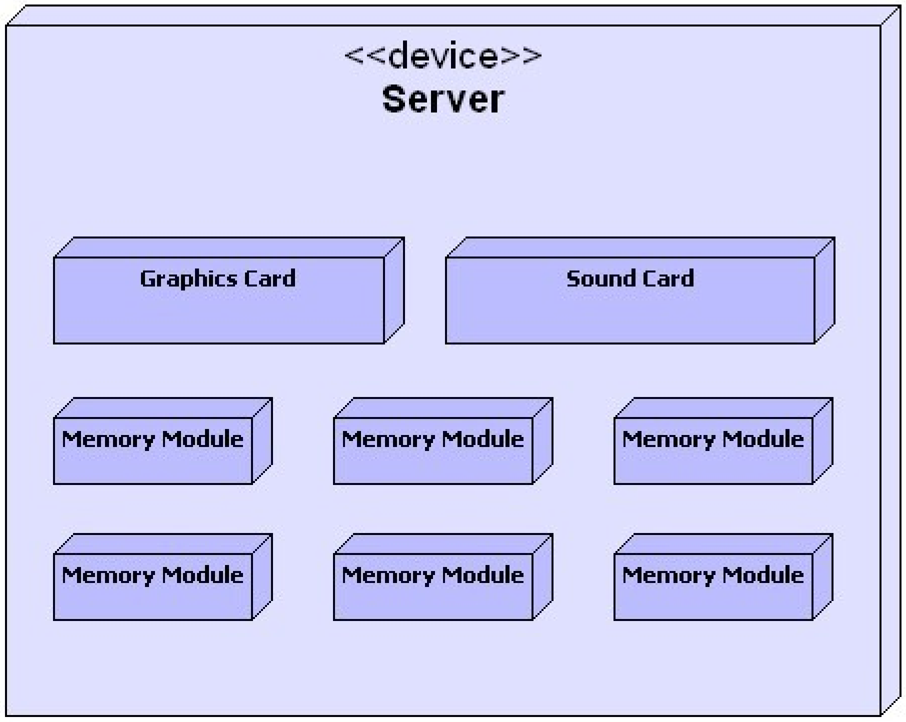
Devices

- A node that represents a physical machine is a “device”.
- By adding the <<device>> stereotype within a node we present a device.



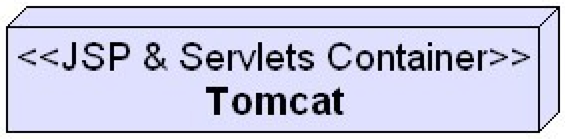
Nested Devices

- A device can be nested within another device. By drawing nested devices it is possible to create a detailed diagram for our system.



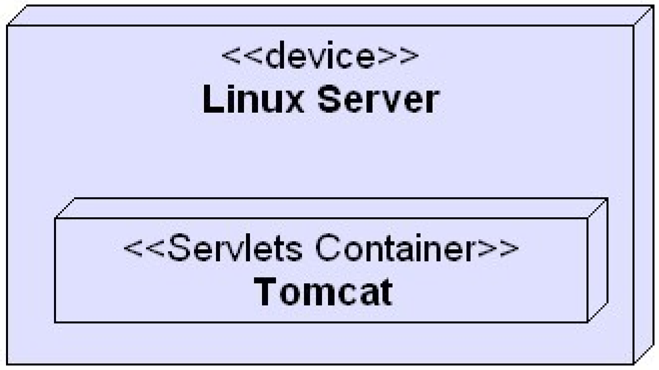
Execution Environment

- A node that represents a software configuration that is capable of running a specific type of artifacts is an “execution environment”.
- By adding the the relevant stereotype within a node we can present that node as an “execution environment” (e.g. `<<Servlets Container>>`).



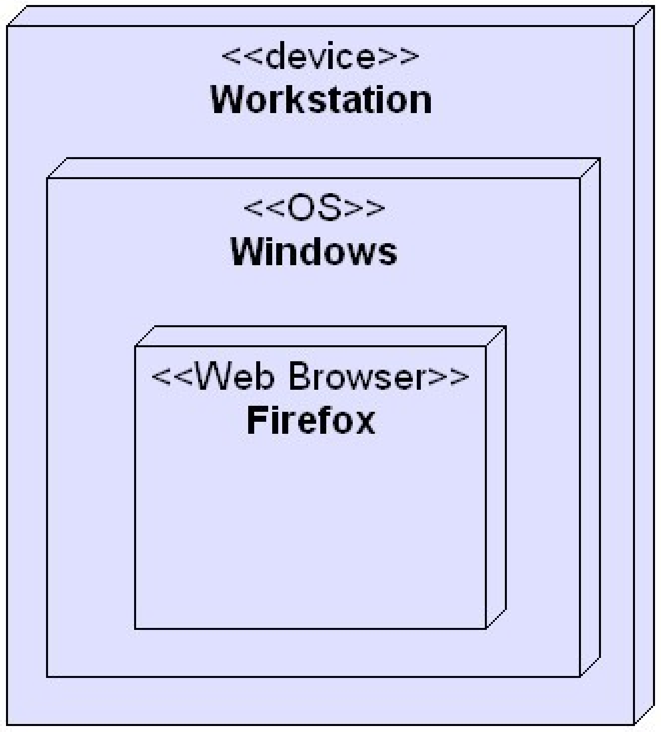
Execution Environment

- The following example presents an execution environment nested within a device.



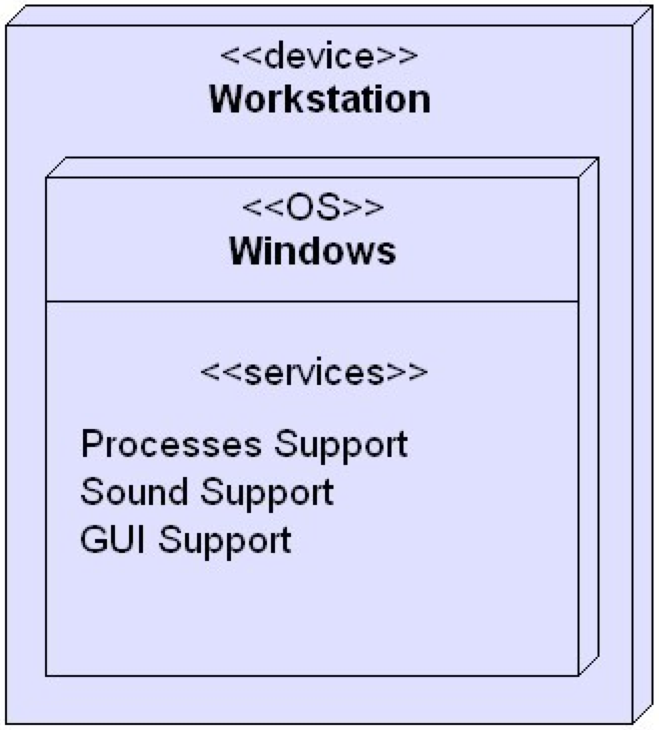
Execution Environment

- The following example presents nested execution environments.



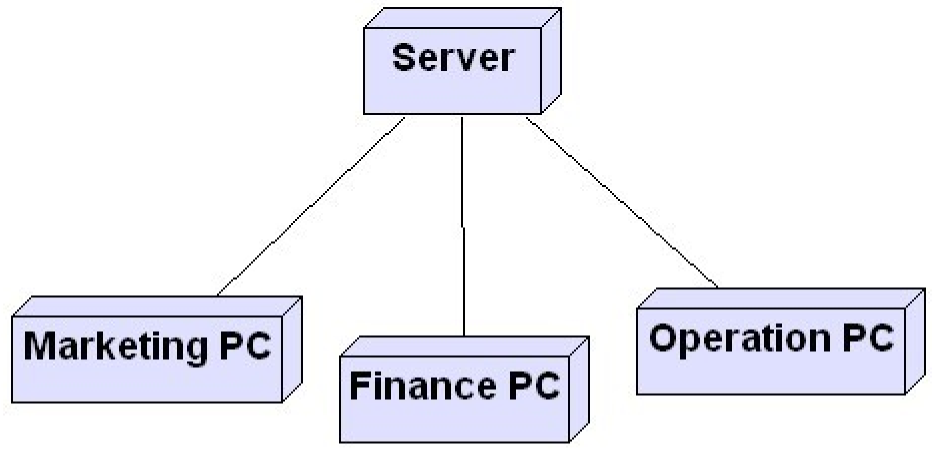
Execution Environment Services

- An execution environment can list its services by adding the `<<services>>` stereotype and listing them below it.



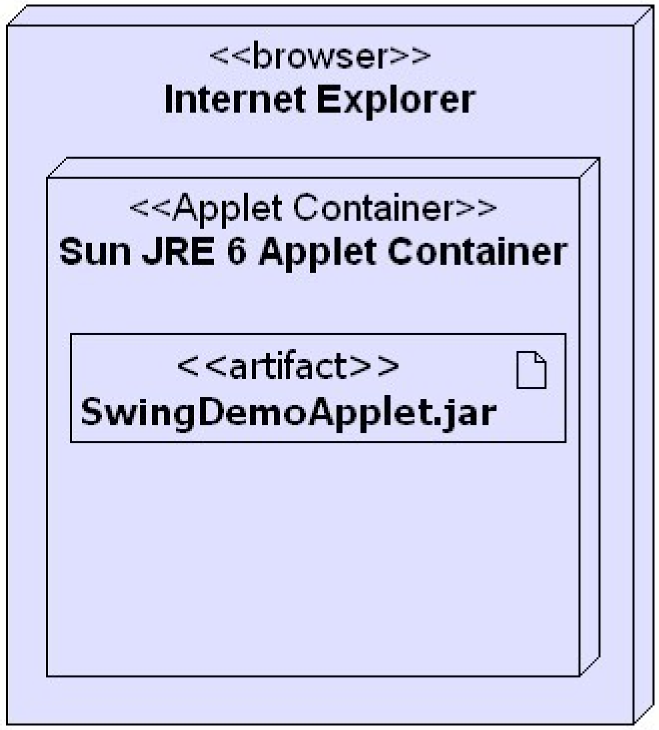
Communication Path

- Generic communication between nodes can be depicted by drawing solid lines from one node to another.



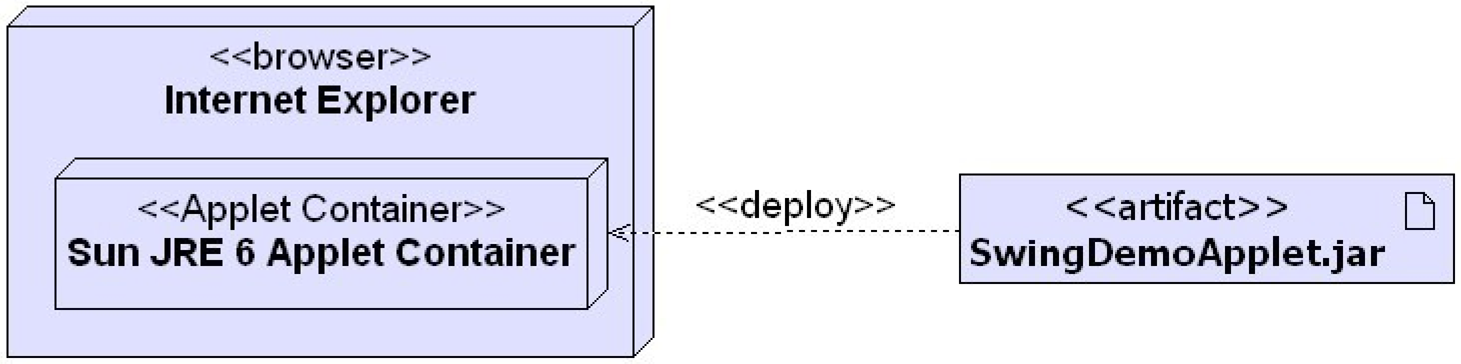
Deployment

- The simplest way to present an artifact deployed on a node is by drawing the artifact within the node's cube.



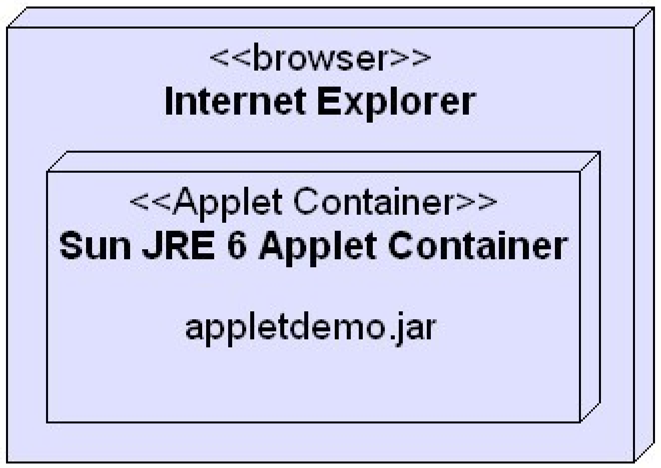
Deployment

- Another way to show deployment includes drawing a dashed line with an open arrow pointing from the artifact to the deployment target. The line should be stereotyped with the `<<deploy>>` key word.



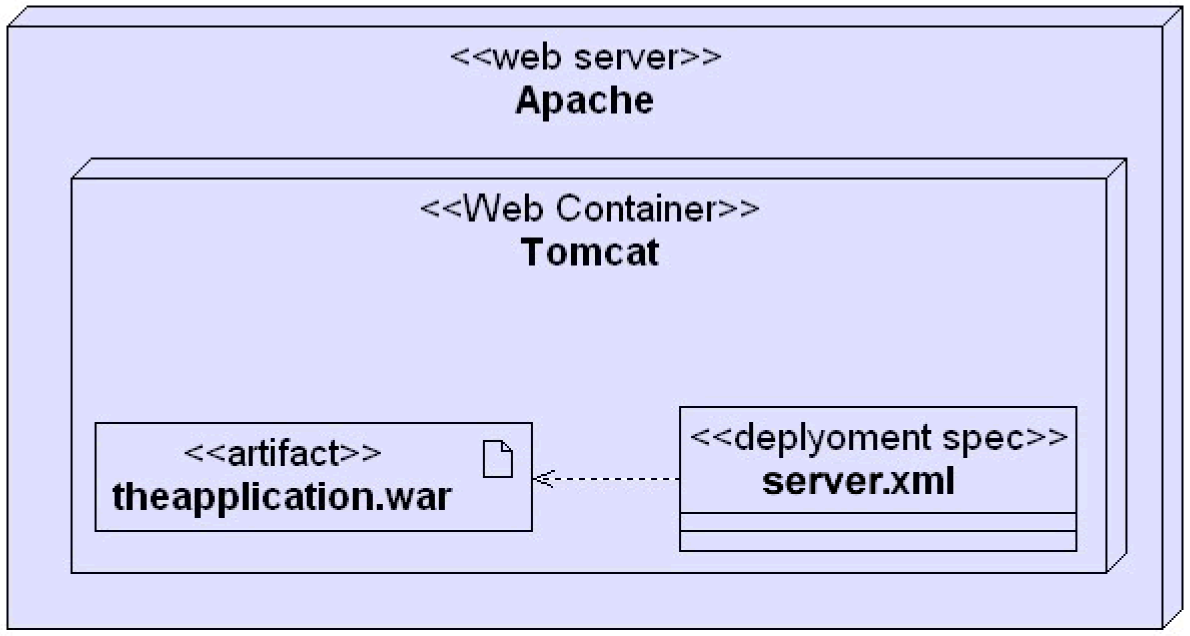
Deployment

- Another way to show deployment includes listing the artifacts within the execution environment on which it is deployed.



Deployment Specifications

- The deployment specifications can be listed as attributes within a class rectangle stereotyped as `<<deployment spec>>`.

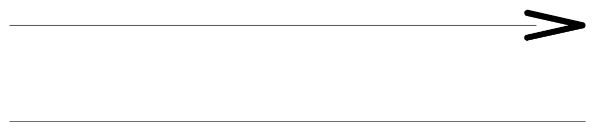


Graphical Paths

- Within a deployment diagram it is possible to use the following graphics paths:

  Association

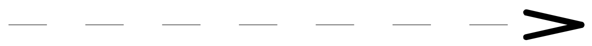
  Using the association lines it is possible to model communication paths between deployment targets.



Graphical Paths

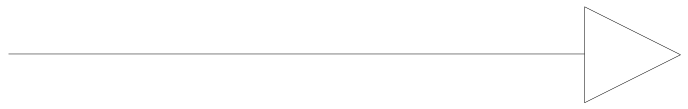
Dependency

Using the dependency line it is possible to show the relationship between an artifact and a node on which it is deployed. Using the dependency line it is also possible to show the dependency between an artifact/node and another model element on which it depends. The other element can be either another artifact or a node.



Generalization

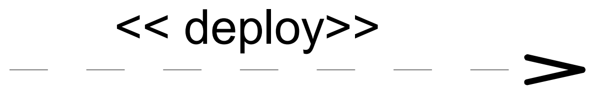
The generalization line allows us showing a generalization relationship between two artifacts and/or two nodes.



Graphical Paths

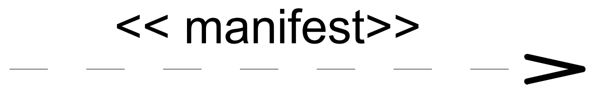
Deployment

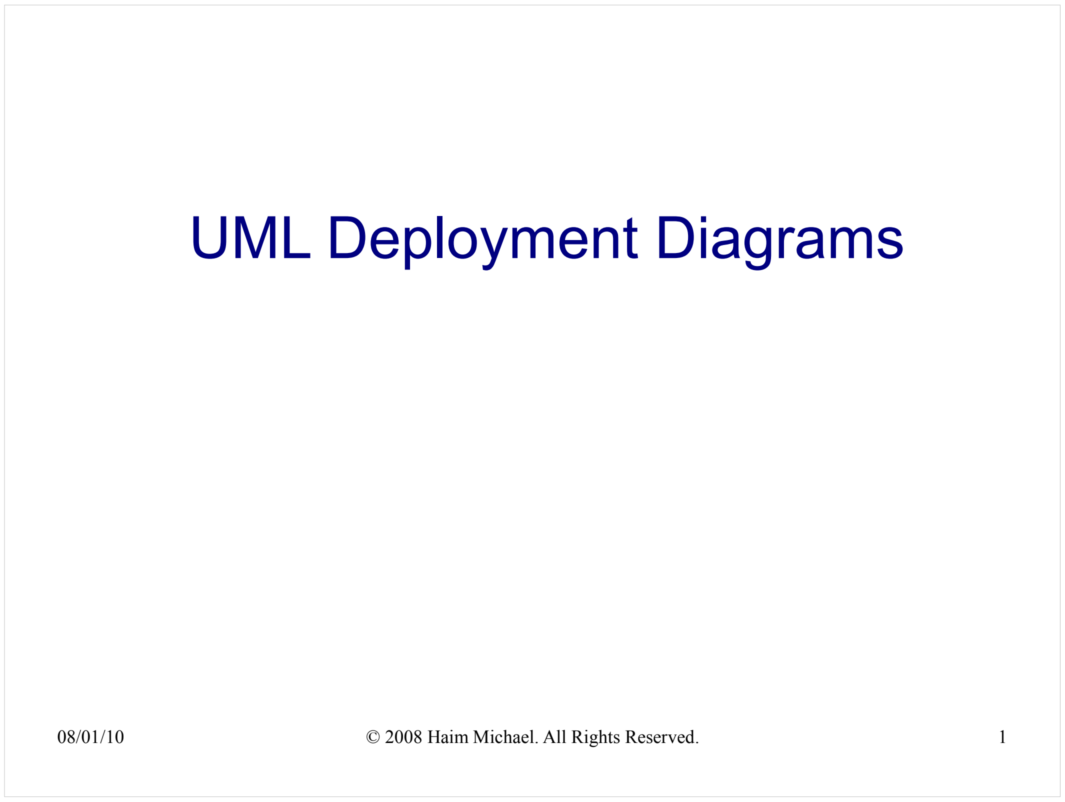
Using the dependency line and the text “<<deploy>>” written above, it is possible to show an artifact deployed on a node.



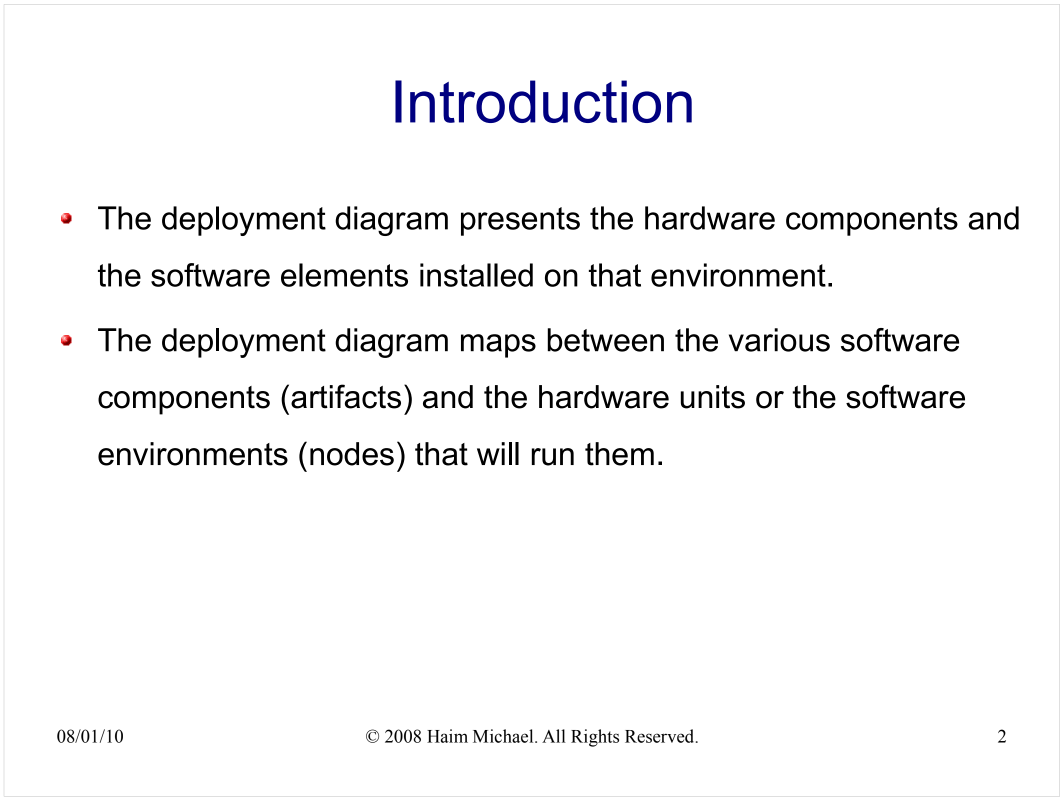
Manifest

Using the dependency line and the text “<<manifest>>” written above, it is possible to show an artifact representing another UML element.



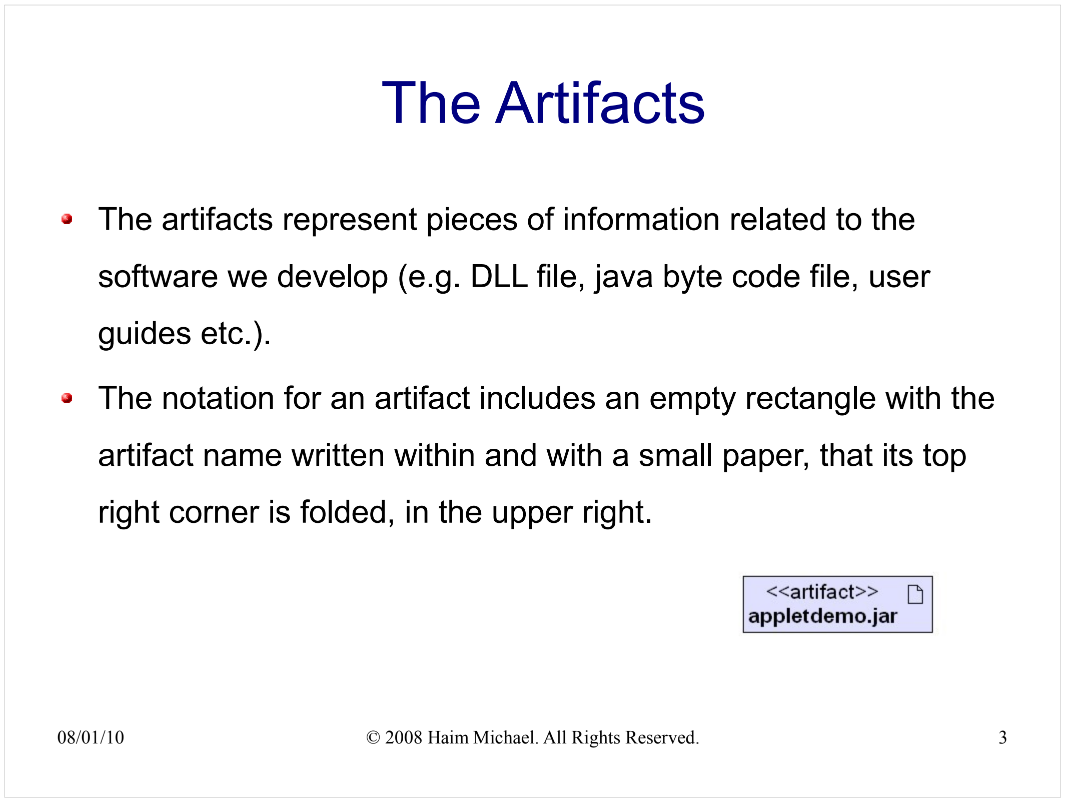


UML Deployment Diagrams



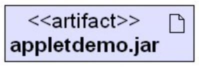
Introduction

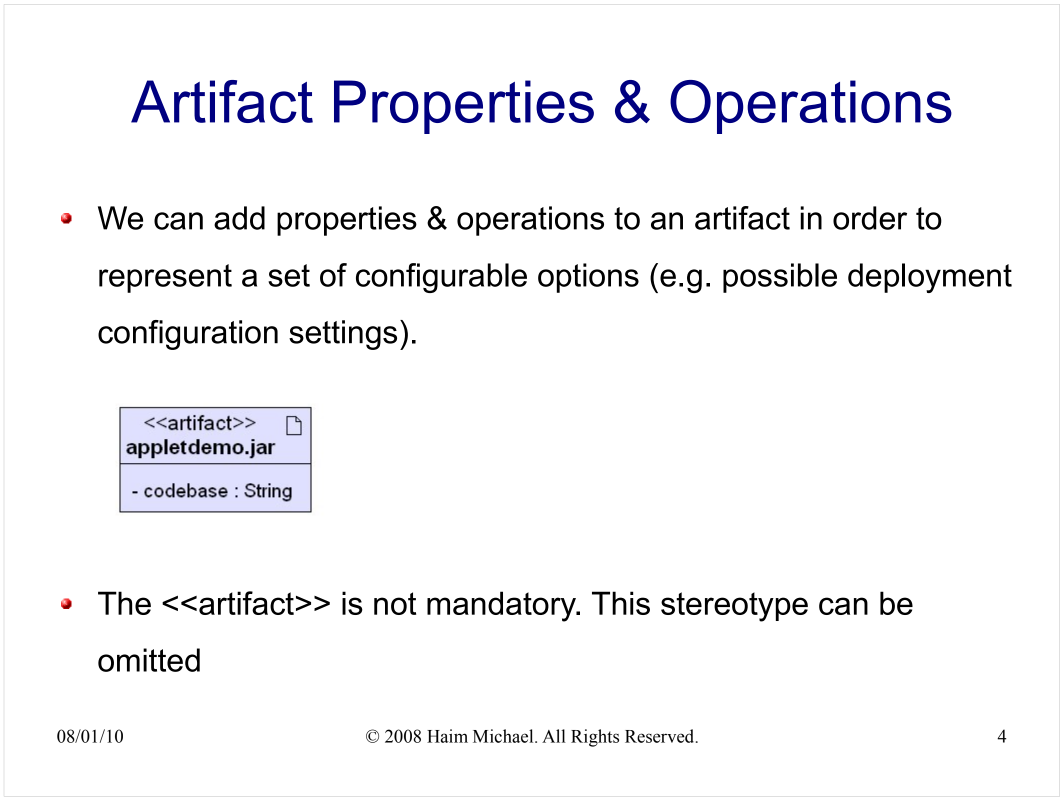
- The deployment diagram presents the hardware components and the software elements installed on that environment.
- The deployment diagram maps between the various software components (artifacts) and the hardware units or the software environments (nodes) that will run them.



The Artifacts

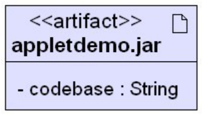
- The artifacts represent pieces of information related to the software we develop (e.g. DLL file, java byte code file, user guides etc.).
- The notation for an artifact includes an empty rectangle with the artifact name written within and with a small paper, that its top right corner is folded, in the upper right.



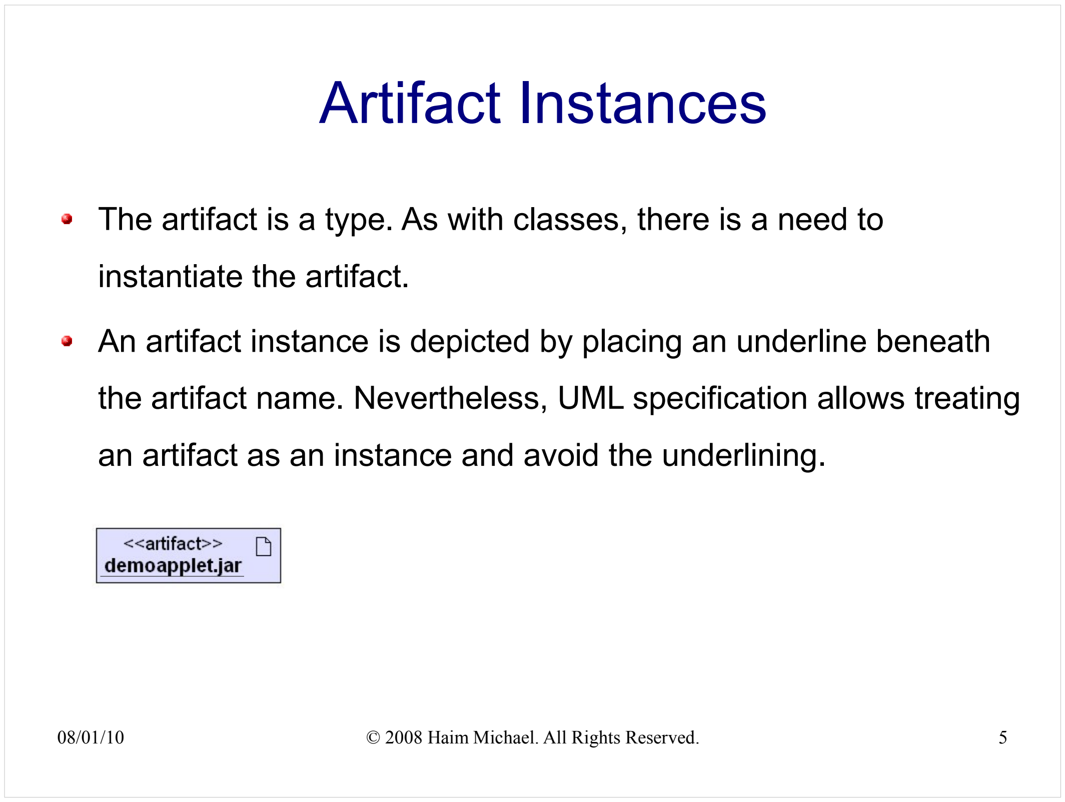


Artifact Properties & Operations

- We can add properties & operations to an artifact in order to represent a set of configurable options (e.g. possible deployment configuration settings).

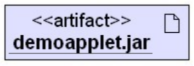


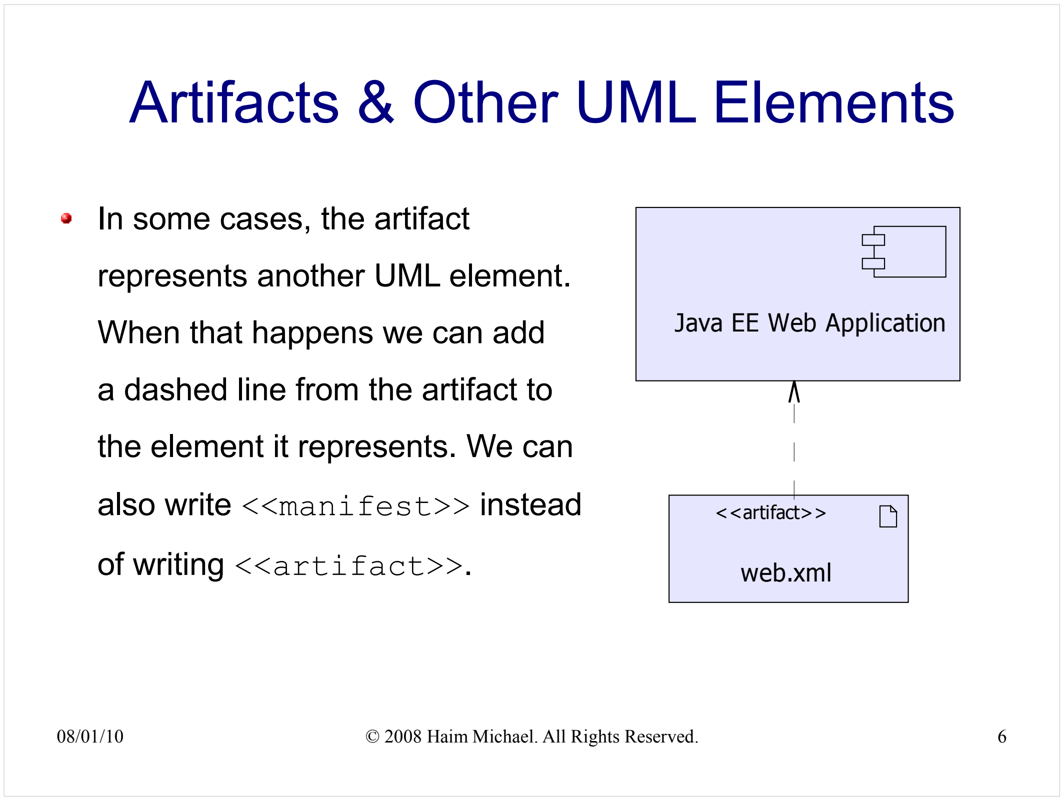
- The <<artifact>> is not mandatory. This stereotype can be omitted



Artifact Instances

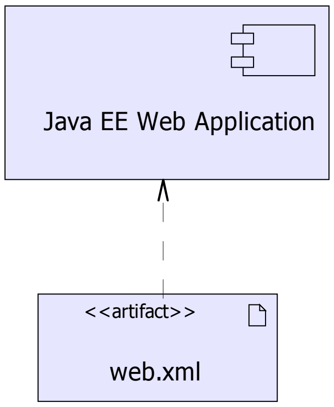
- The artifact is a type. As with classes, there is a need to instantiate the artifact.
- An artifact instance is depicted by placing an underline beneath the artifact name. Nevertheless, UML specification allows treating an artifact as an instance and avoid the underlining.

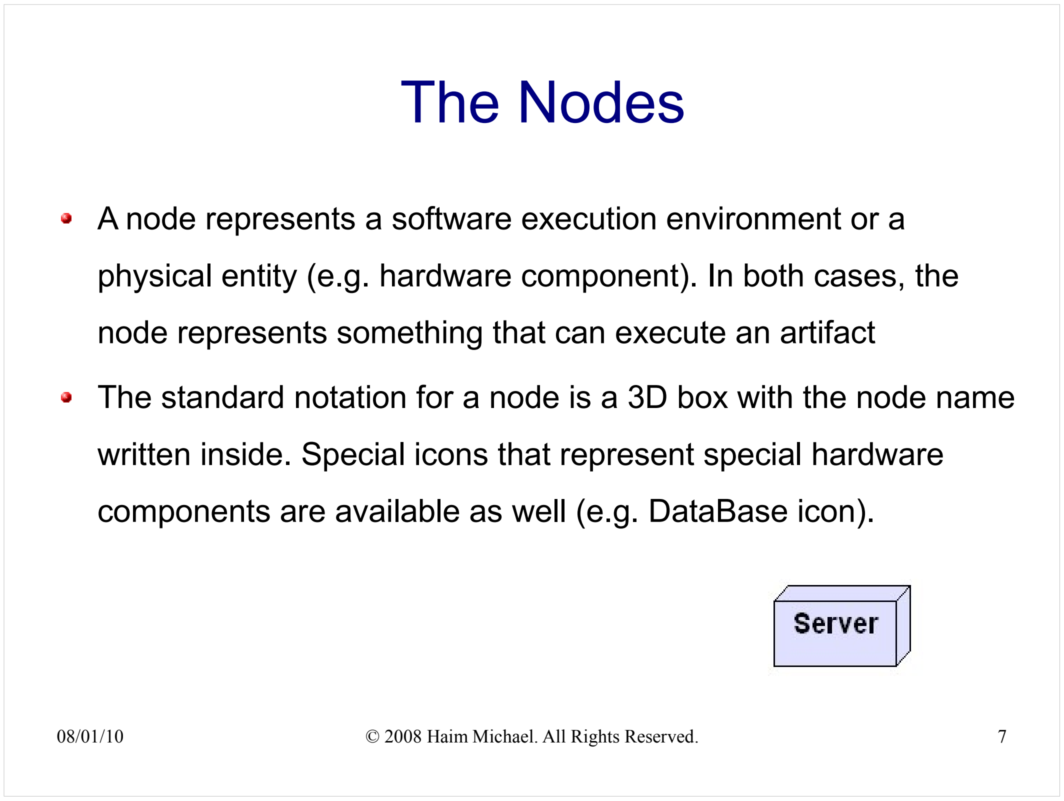




Artifacts & Other UML Elements

- In some cases, the artifact represents another UML element. When that happens we can add a dashed line from the artifact to the element it represents. We can also write `<<manifest>>` instead of writing `<<artifact>>`.





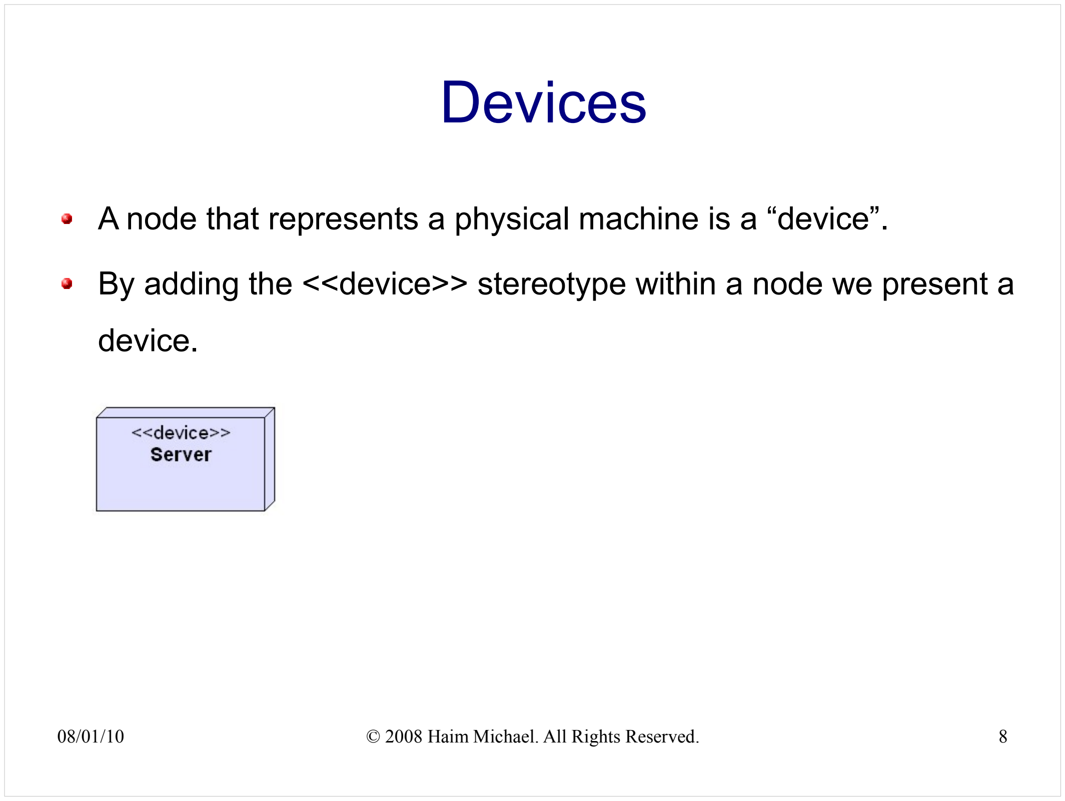
The Nodes

- A node represents a software execution environment or a physical entity (e.g. hardware component). In both cases, the node represents something that can execute an artifact
- The standard notation for a node is a 3D box with the node name written inside. Special icons that represent special hardware components are available as well (e.g. DataBase icon).

08/01/10 © 2008 Haim Michael. All Rights Reserved. 7

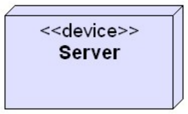
As with Artifacts, it is common to treat the Node as a class we need to instantiate, and depict each instance the same way we depict the Node, only with an underline.

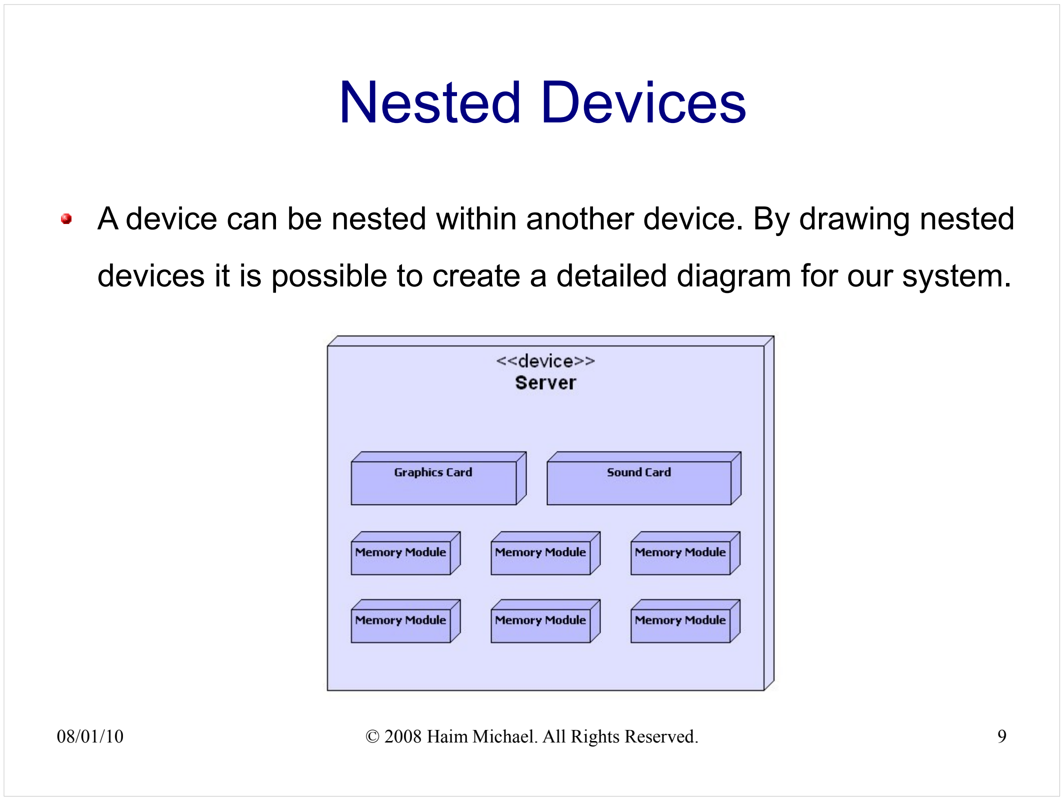
As with Artifacts, the underline can be omitted.



Devices

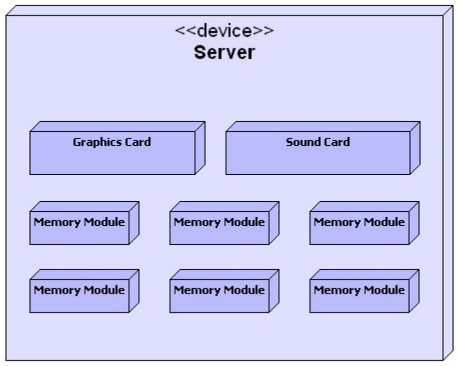
- A node that represents a physical machine is a “device”.
- By adding the <<device>> stereotype within a node we present a device.





Nested Devices

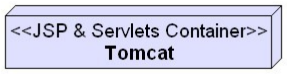
- A device can be nested within another device. By drawing nested devices it is possible to create a detailed diagram for our system.

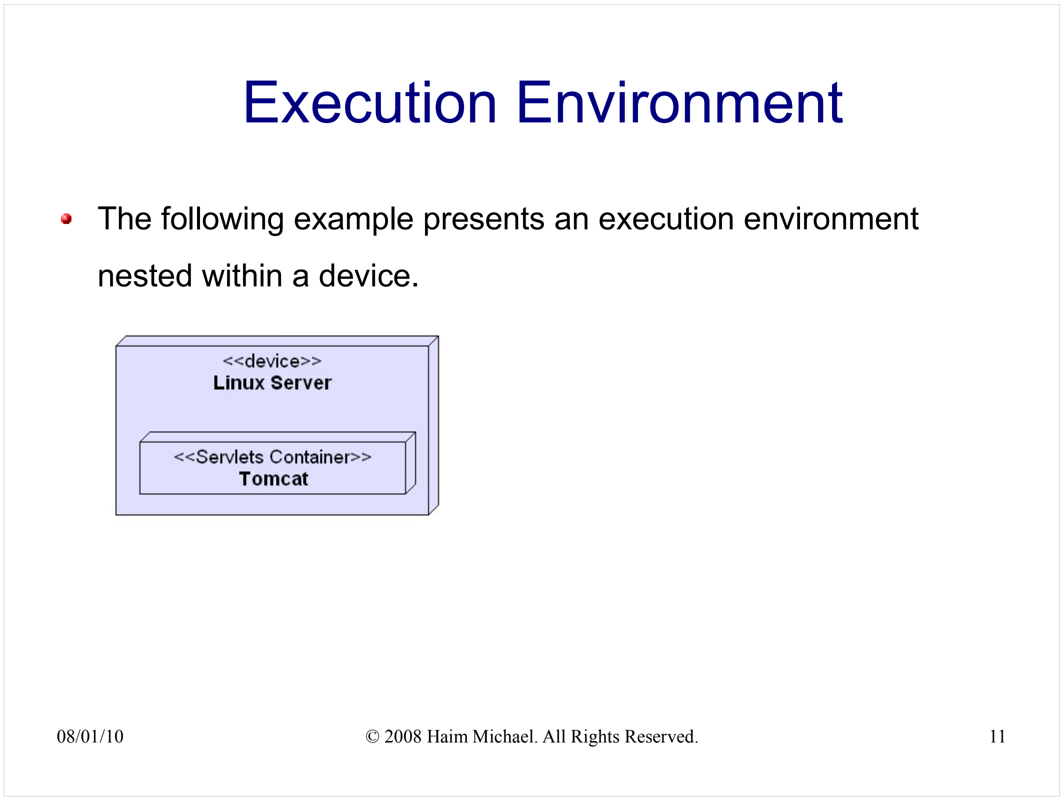


08/01/10 © 2008 Haim Michael. All Rights Reserved. 9

Execution Environment

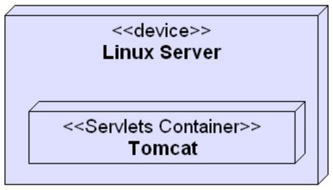
- A node that represents a software configuration that is capable of running a specific type of artifacts is an “execution environment”.
- By adding the the relevant stereotype within a node we can present that node as an “execution environment” (e.g. `<<Servlets Container>>`).





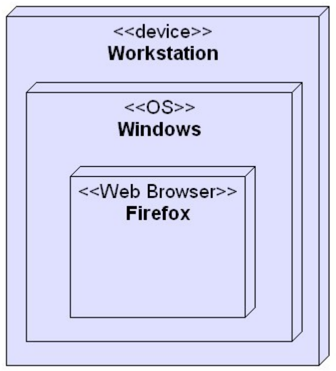
Execution Environment

- The following example presents an execution environment nested within a device.



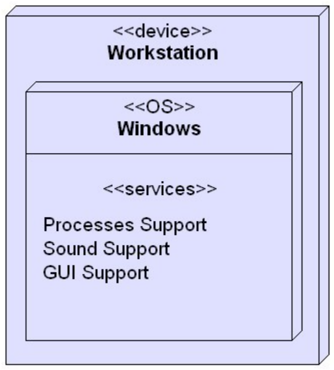
Execution Environment

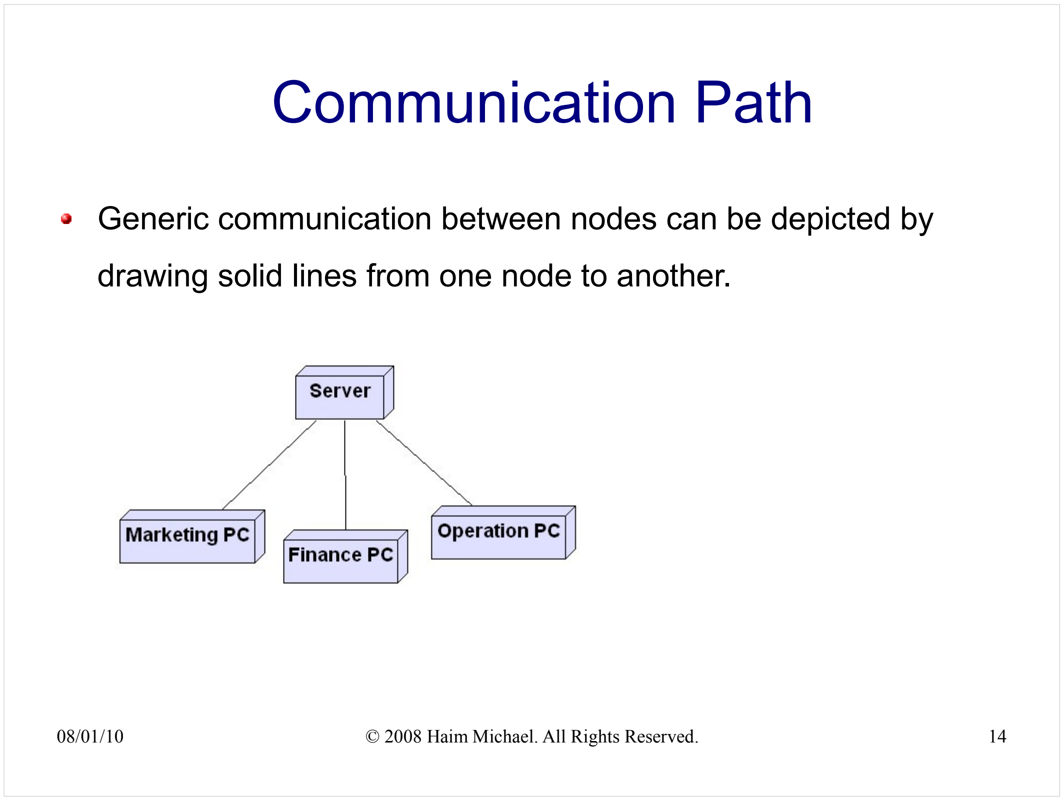
- The following example presents nested execution environments.



Execution Environment Services

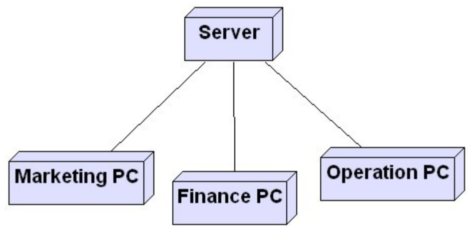
- An execution environment can list its services by adding the `<<services>>` stereotype and listing them below it.



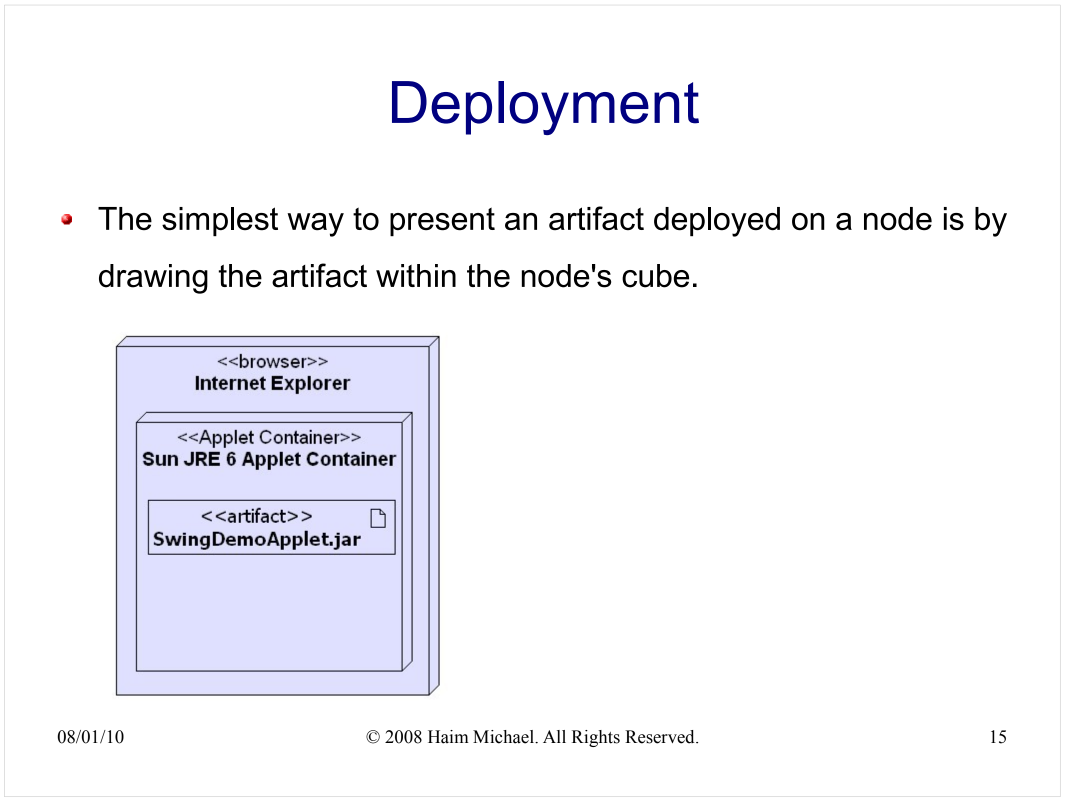


Communication Path

- Generic communication between nodes can be depicted by drawing solid lines from one node to another.

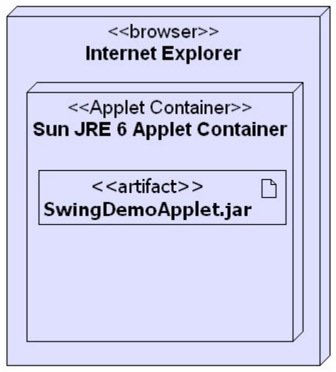


08/01/10 © 2008 Haim Michael. All Rights Reserved. 14

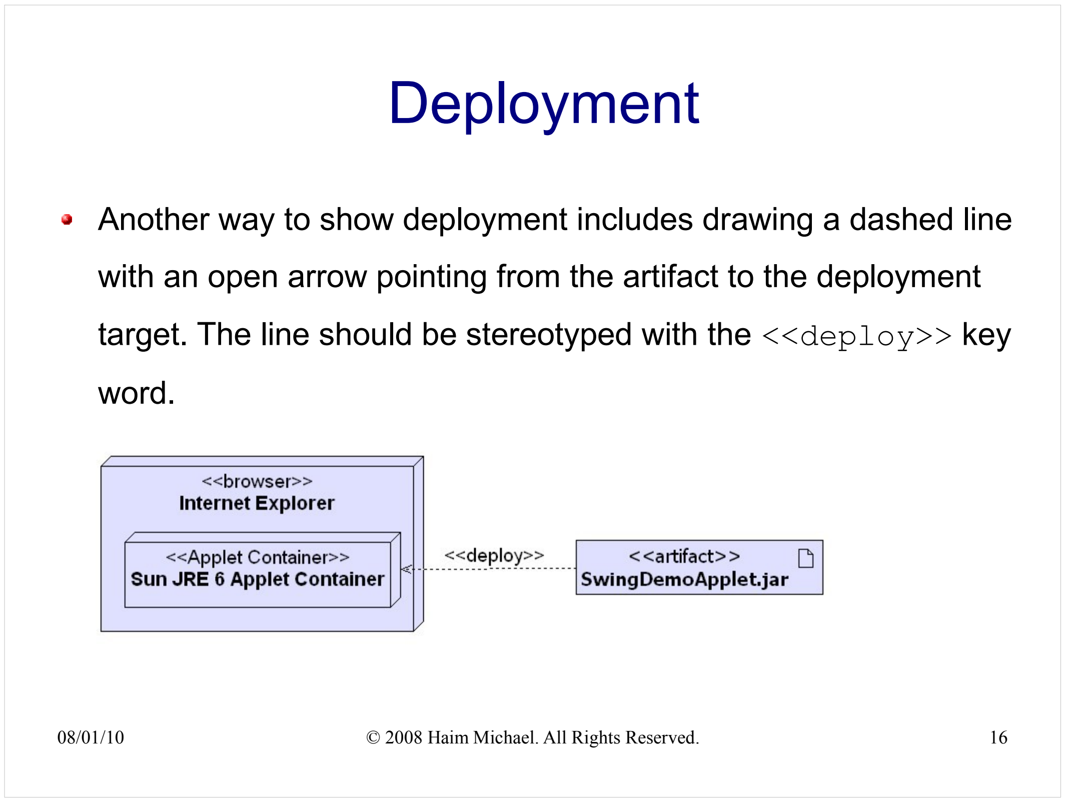


Deployment

- The simplest way to present an artifact deployed on a node is by drawing the artifact within the node's cube.

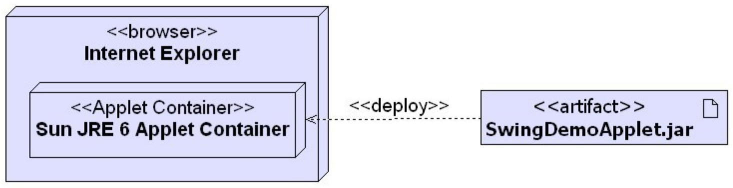


08/01/10 © 2008 Haim Michael. All Rights Reserved. 15



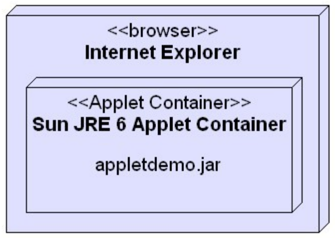
Deployment

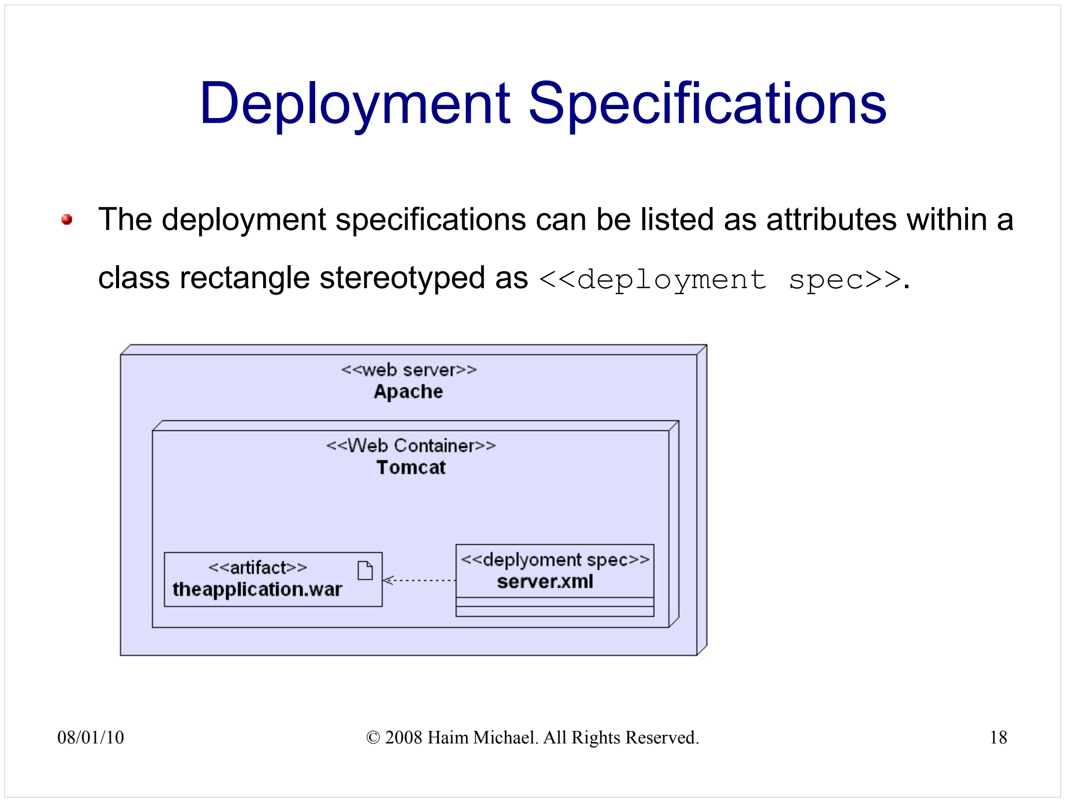
- Another way to show deployment includes drawing a dashed line with an open arrow pointing from the artifact to the deployment target. The line should be stereotyped with the `<<deploy>>` key word.



Deployment

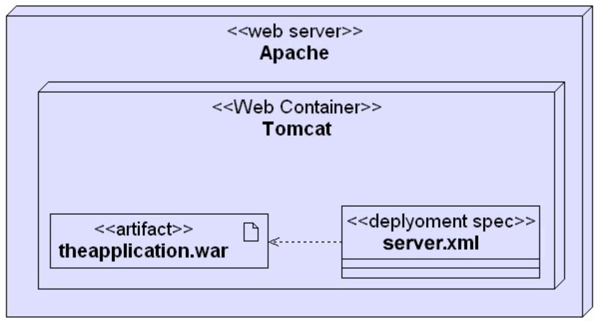
- Another way to show deployment includes listing the artifacts within the execution environment on which it is deployed.

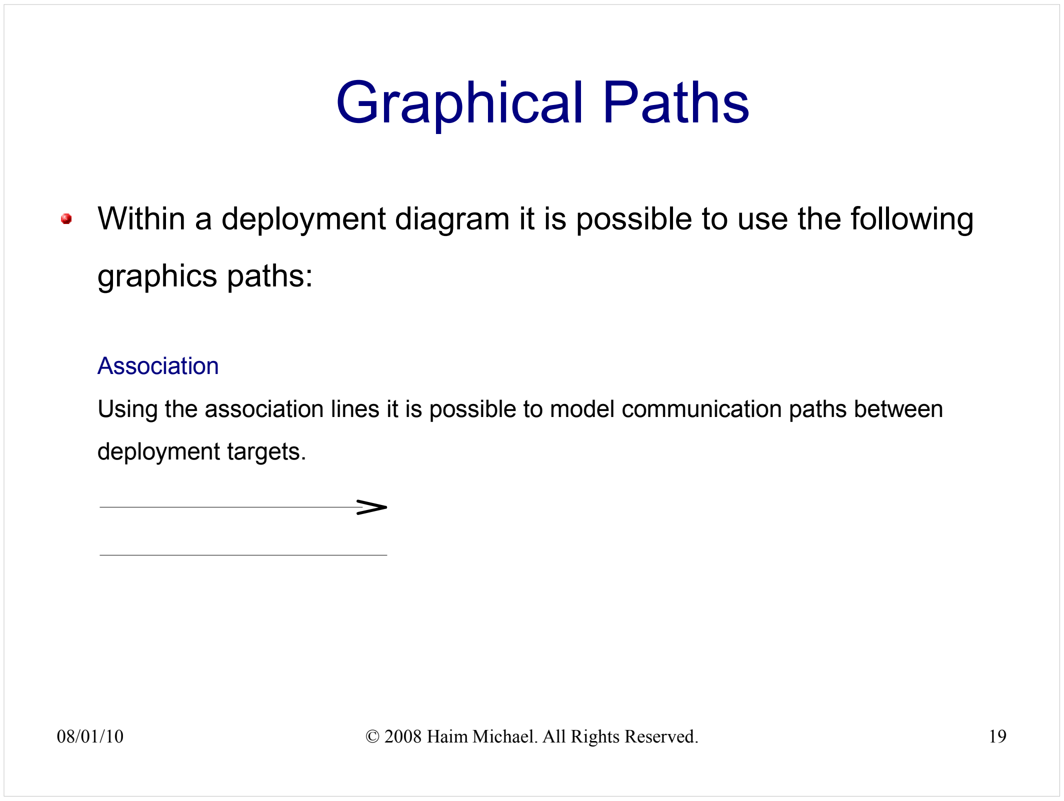




Deployment Specifications

- The deployment specifications can be listed as attributes within a class rectangle stereotyped as `<<deployment spec>>`.



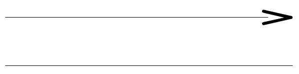


Graphical Paths

- Within a deployment diagram it is possible to use the following graphics paths:

Association

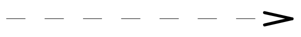
Using the association lines it is possible to model communication paths between deployment targets.



Graphical Paths

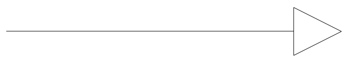
Dependency

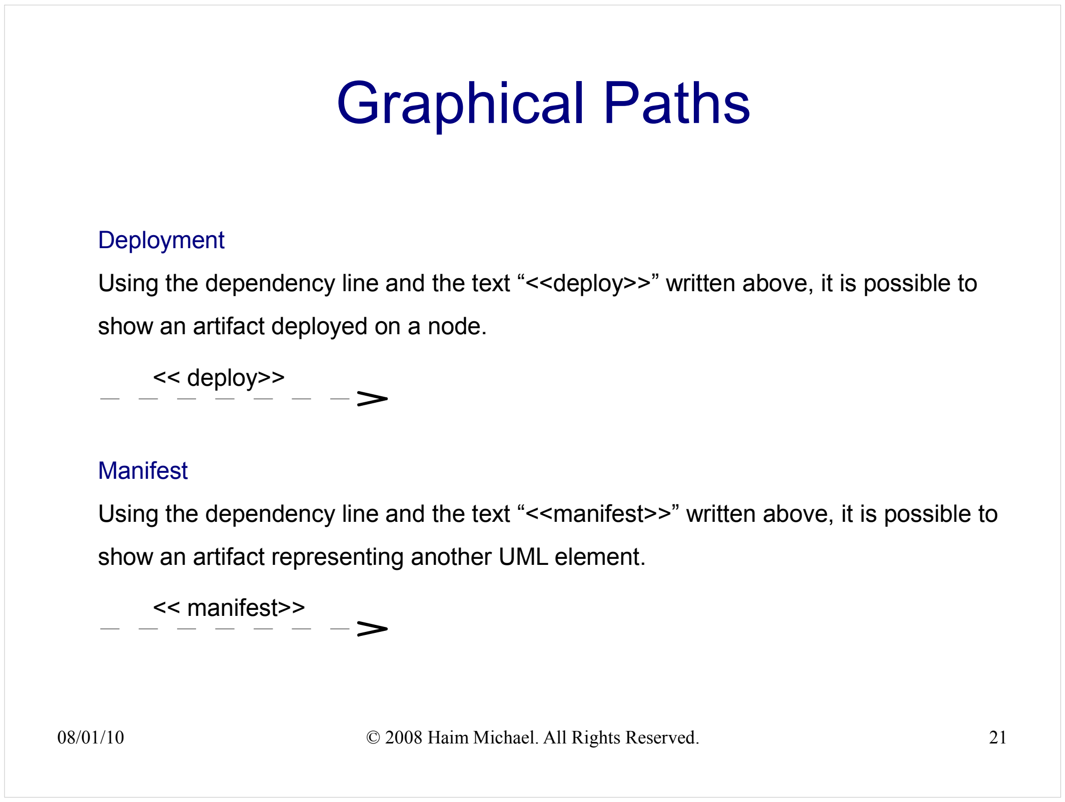
Using the dependency line it is possible to show the relationship between an artifact and a node on which it is deployed. Using the dependency line it is also possible to show the dependency between an artifact/node and another model element on which it depends. The other element can be either another artifact or a node.



Generalization

The generalization line allows us showing a generalization relationship between two artifacts and/or two nodes.





Graphical Paths

Deployment

Using the dependency line and the text “<<deploy>>” written above, it is possible to show an artifact deployed on a node.

<< deploy>>
– – – – – – – >

Manifest

Using the dependency line and the text “<<manifest>>” written above, it is possible to show an artifact representing another UML element.

<< manifest>>
– – – – – – – >

08/01/10 © 2008 Haim Michael. All Rights Reserved. 21