

---

**Enterprise Architect**

# **The Physical Model**

*1.0*



Geoffrey Sparks

# Contents

<b>Introduction</b>	<b>3</b>
<hr/>	
<b>The UML</b>	<b>3</b>
<hr/>	
<b>The Deployment Model</b>	<b>4</b>
<hr/>	
<b>Node Instances</b>	<b>5</b>
<hr/>	
<b>Components</b>	<b>6</b>
<hr/>	
<b>Communication</b>	<b>7</b>
<hr/>	
<b>Dependencies</b>	<b>8</b>
<hr/>	
<b>Business Modelling and Implementation Diagrams</b>	<b>9</b>
<hr/>	
<b>Suggested Reading</b>	<b>10</b>
<hr/>	

## ***Introduction***

The Physical Model in UML describes the components, both hardware and software, that will be deployed into the target environment. It describes such items as hardware platforms, called 'Nodes' in UML, network connectivity, software components, processors, operating systems and 3rd party tools.

Deployment diagrams are the complement of Component diagrams, and these together provide the Implementation View of the system. This paper will look out the notation used in Deployment diagrams and some examples of how they are used.

## ***The UML***

The Unified Modelling Language (UML) is, as its name implies, a modelling language and not a method or process. UML is made up of a very specific notation and the related grammatical rules for constructing software models. UML in itself does not proscribe or advise on how to use that notation in a software development process or as part of an object-oriented design methodology.

UML supports a rich set of graphical notation elements. It describes the notation for classes, components, nodes, activities, work flow, use cases, objects, states and how to model relationships between these elements. UML also supports the notion of custom extensions through stereotyped elements.

The UML provides significant benefits to software engineers and organisations by helping to build rigorous, traceable and maintainable models, which support the full software development lifecycle.

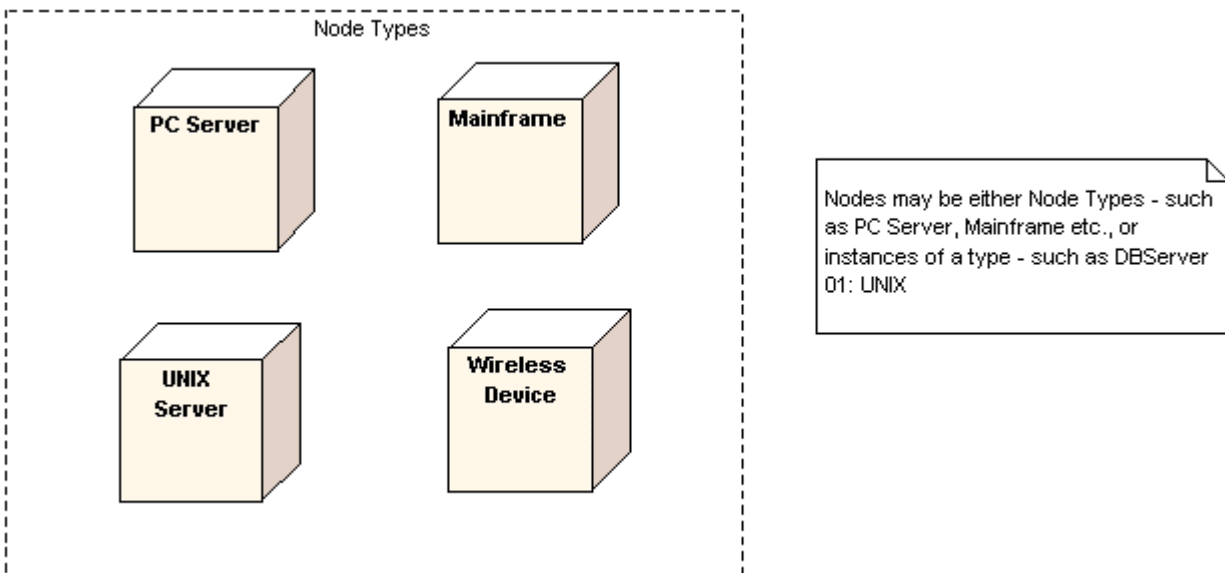
You can find out more about UML from the books mentioned in the suggested reading section and from the UML specification documents to be found at the Object Management Groups UML resource pages:

<http://www.omg.org/technology/uml/> and at  
<http://www.omg.org/technology/documents/formal/>

## The Deployment Model

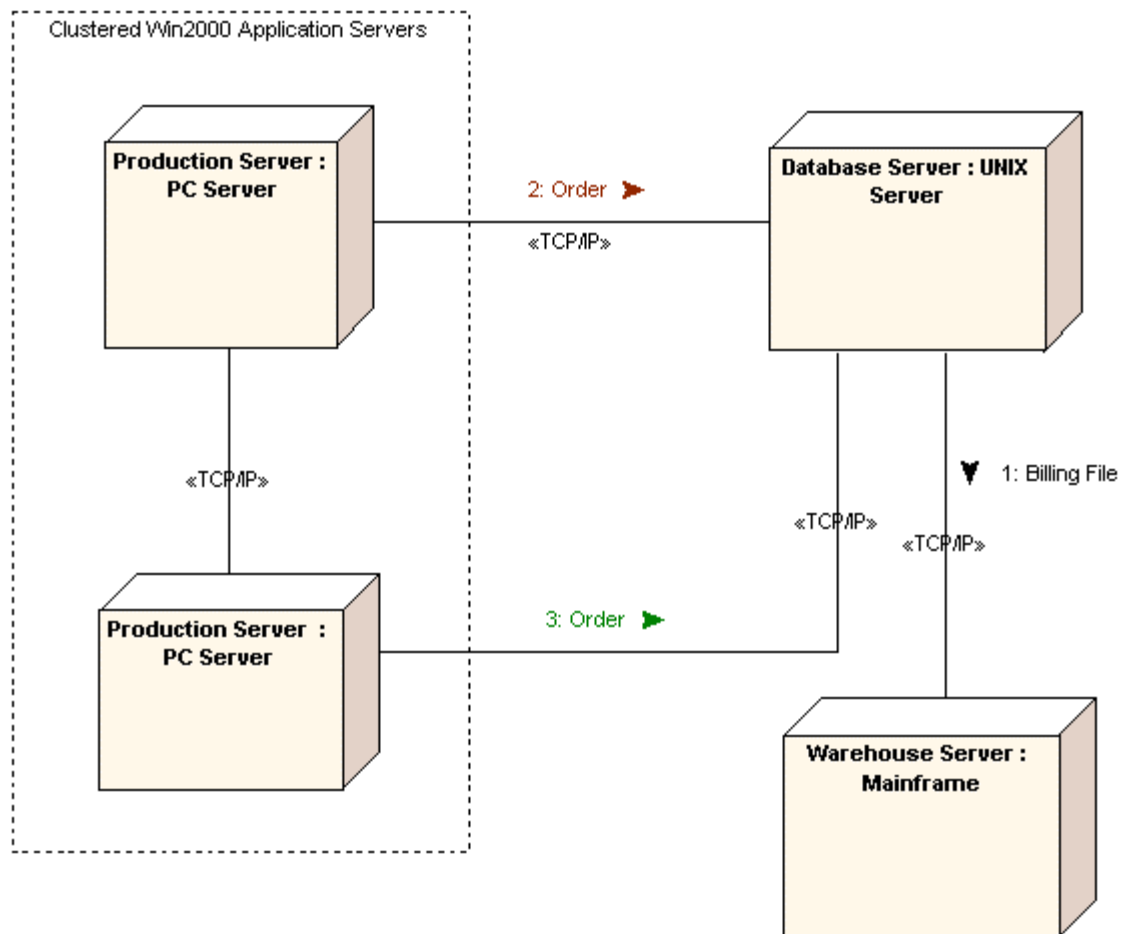
Deployment diagrams depict Nodes and their relationships. Typically, Nodes are connected by communication associations - such as network links, TCP-IP connections, microwave & etc.

The image below shows a variety of Node Types. These are abstract types that may be implemented at runtime by physical instances. In much the same manner as a Class and an Object are used to model the abstract definition and the run-time instance.



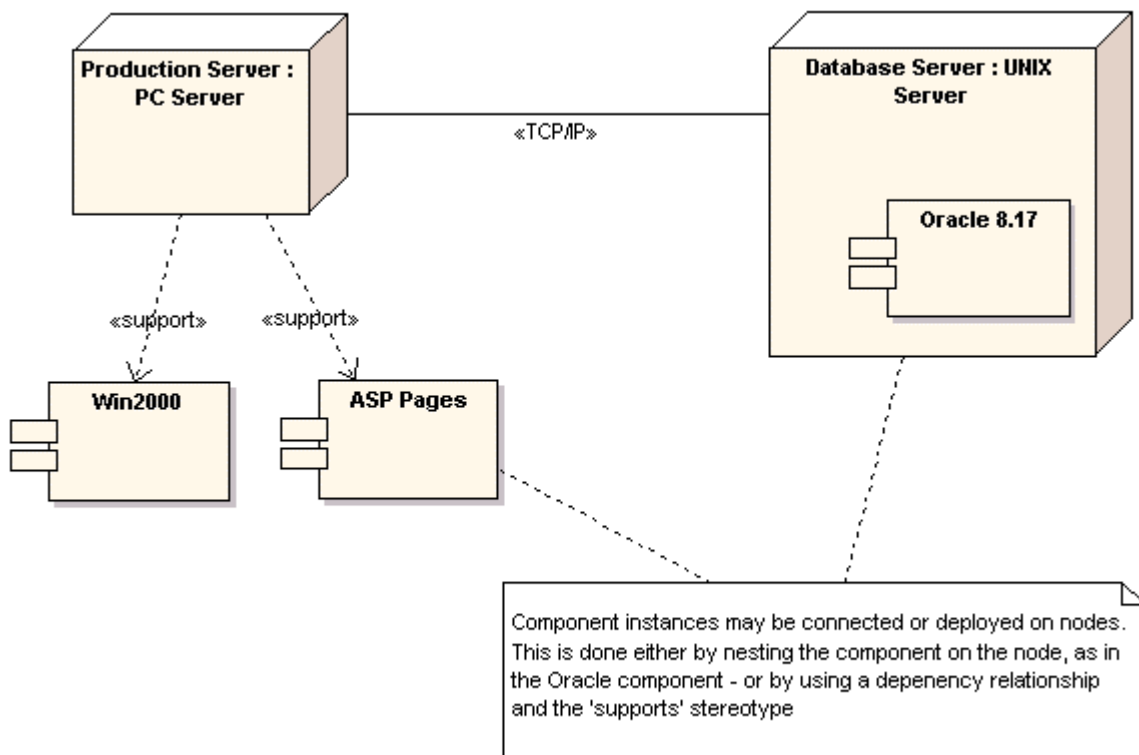
## Node Instances

Once you have defined Node Types, you can create the equivalent of a Collaboration diagram, using instances of Nodes as they will be deployed at run time. The example below shows how instances of the Node Types already defined can be created and linked together with communication associations. Collaboration messages have also been used to depict the flow of information or discrete objects between nodes, such as the billing file and order information.



## Components

Component instances are used to illustrate the executables, applications and other software components that are deployed on nodes. The diagram below illustrates this:

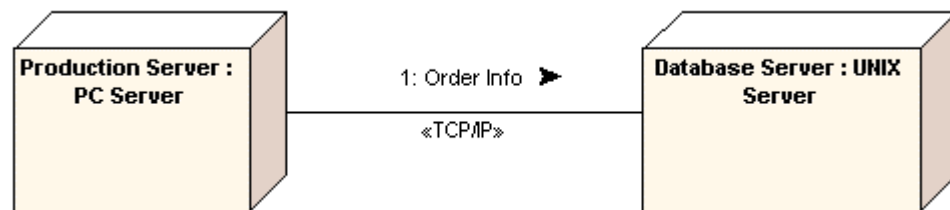


An instance is a runtime physical example of an abstract type

## Communication

Communication between nodes is depicted using associations, which may be stereotyped to illustrate the exact protocol used to communicate - such as TCP/IP or SNA.

The image below shows two node instances - a PC Server named "Production Server" and a UNIX Server named "Database Server". These two instances are connected by a network link running TCP/IP. One of the main purposes of this link is to post Order information from the production application server to the database, so a collaboration message has been added to model this usage.

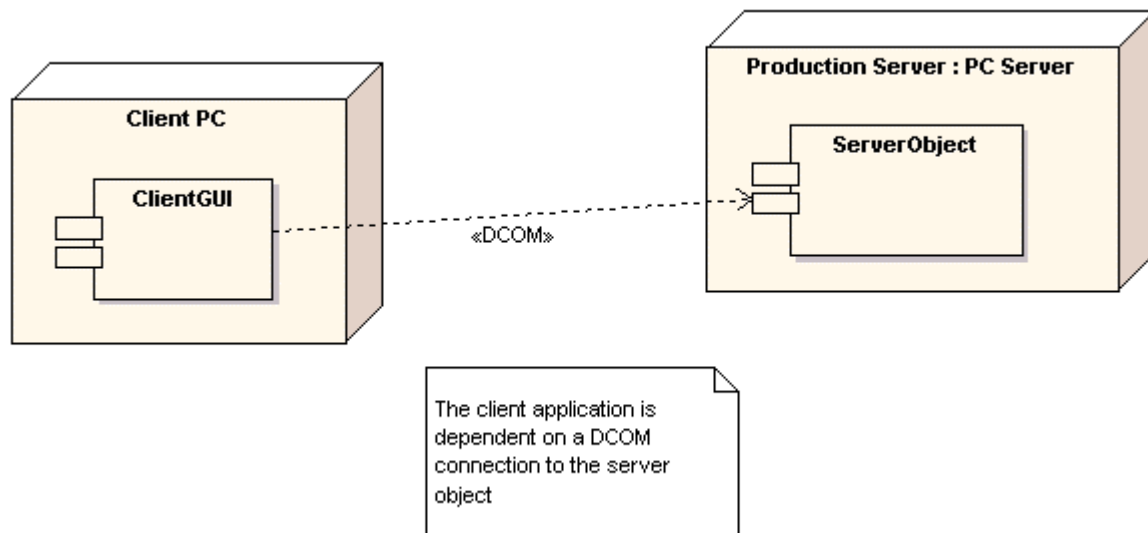


Communication between nodes can take many forms, for example

- Network connectivity
- TCP/IP
- SNA
- Microwave
- Infrared
- Wireless Protocol

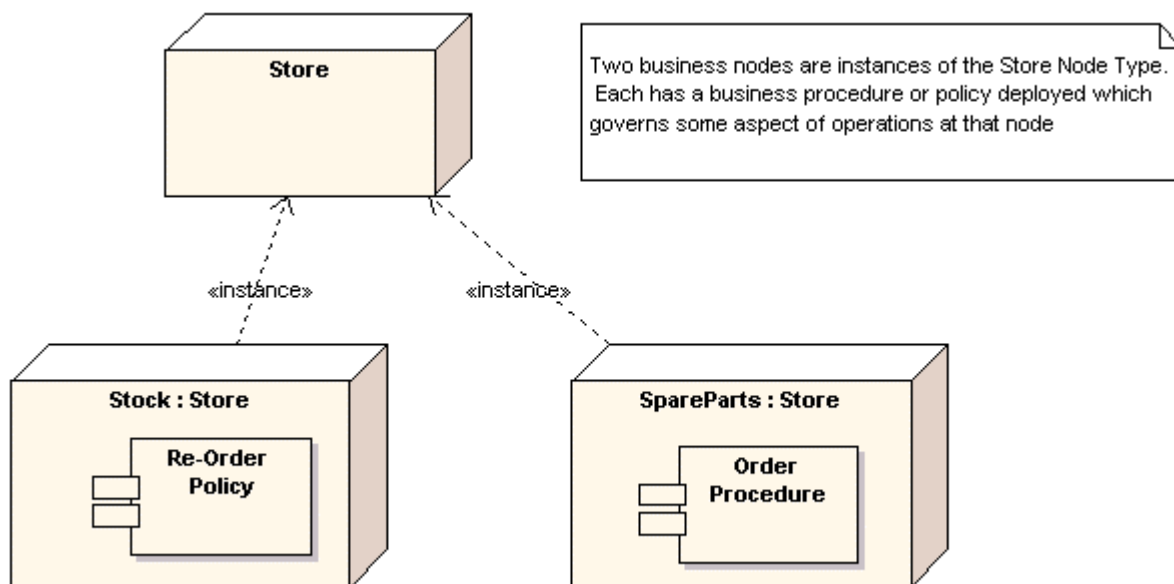
## Dependencies

Dependencies are used in Deployment Diagrams to illustrate the dependent relationships between artifacts. These may be stereotyped to indicate the actual relationship - for example <<DCOM>> may be indicated as a dependent distributed protocol for connecting a client and a server object.



## Business Modelling and Implementation Diagrams

Deployment diagrams can also model non-computer entities and systems within an organisation - such as the StoreRoom, FrontCounter, MainOffice and SparePartsDivision. Business components can be used to model things such as business rules, procedures and documents. These components can then be deployed onto business nodes. The example below illustrates this idea:



---

Business Modelling using nodes and components is an effective means of capturing non-computer based processes and entities. This can be done very early in the analysis phase to complement the use case model and other business modelling.

---

## ***Suggested Reading***

**Sinan Si Alhir, UML in a NutShel.**

ISBN: 1-56592-448-7. Publisher: O'Reilly & Associates, Inc

**Doug Rosenberg with Kendall Scott ,User Interface Driven Object  
Modeling with UML.**

ISBN:0-201-43289-7. Publisher: Addison-Wesley

**Geri Scheider, Jason P. Winters, Applying User Interfaces**

ISBN: 0-201-30981-5. Publisher: Addison-Wesley

**Ivar Jacobson, Martin Griss, Patrik Jonsson, Software Reuse**

ISBN:0-201-92476-5. Publisher: Addison-Wesley

**Hans-Erik Eriksson, Magnus Penker, Business Modeling with UML**

ISBN: 0-471-29551-5. Publisher: John Wiley & Son, Inc

**Peter Herzum, Oliver Sims, Business Component Factory**

ISBN: 0-471-32760-3 Publisher: John Wiley & Son, Inc