

# An Introduction to UML

# The Component Model

by Geoffrey Sparks

*All material (c) Geoffrey Sparks 2000*

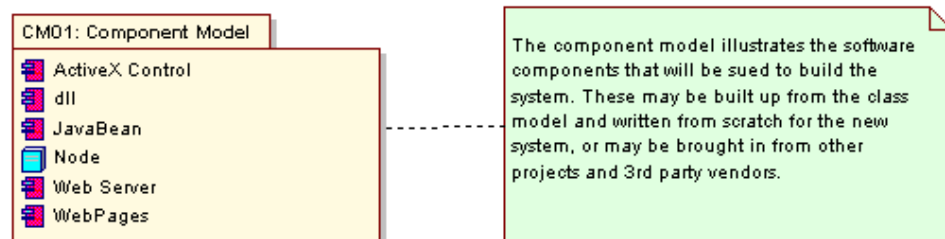
[www.sparxsystems.com.au](http://www.sparxsystems.com.au)

## Table of Contents

<b>THE COMPONENT MODEL .....</b>	<b>3</b>
INTRODUCTION TO UML.....	3
COMPONENT NOTATION .....	4
<i>The Component Diagram</i> .....	4
<i>Interfaces</i> .....	5
<i>Components and Nodes</i> .....	5
<i>Requirements</i> .....	6
<i>Constraints</i> .....	7
<i>Scenarios</i> .....	8
TRACEABILITY .....	9
AN EXAMPLE .....	10
<i>Server Components</i> .....	10
<i>Security Components</i> .....	11
<i>Recommended Reading</i> .....	12

## The Component Model

This paper describes how to model software and hardware components in the UML. The component model illustrates the software components that will be used to build the system. These may be built up from the class model and written from scratch for the new system, or may be brought in from other projects and 3rd party vendors. Components are high level aggregations of smaller software pieces, and provide a 'black box' building block approach to software construction.



### *Introduction to 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, Components, objects, states and how to model relationships between these elements. UML also supports the notion of custom extensions through stereotyped elements.

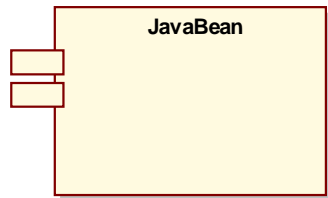
The UML provides significant benefits to software engineers and organisations by helping to build rigorous, traceable and maintainable models, which support the full software development lifecycle.

This paper focuses on ...

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/>

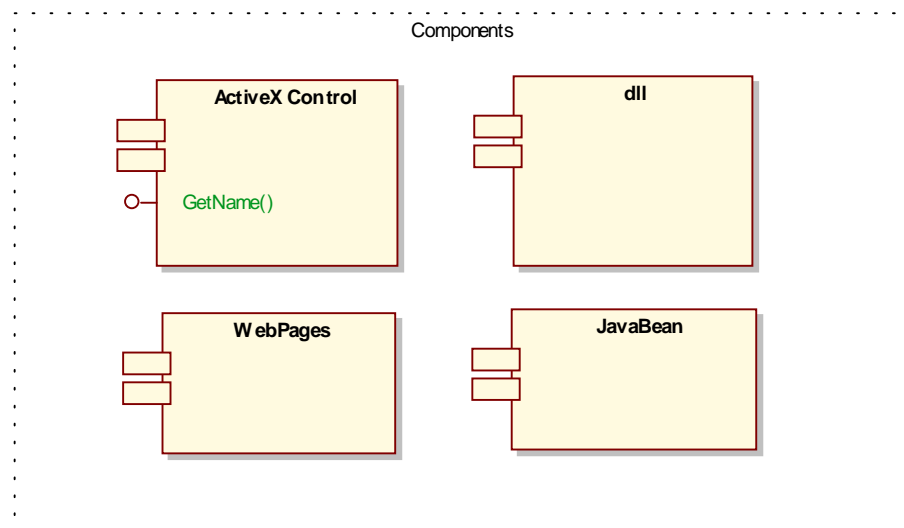
## Component Notation

A component may be something like an ActiveX control - either a user interface control or a business rules server. Components are drawn as the following diagram shows:



## The Component Diagram

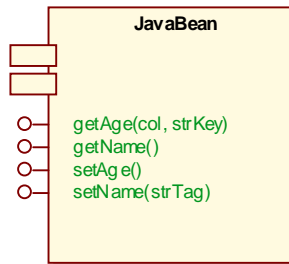
The component diagram shows the relationship between software components, their dependencies, communication, location and other conditions.



Components are the building blocks which make up the deployed system. Typically they are high level artifacts such as java beans, .dll's, ocx's and other software products, often made up of simpler elements such as classes and function libraries.

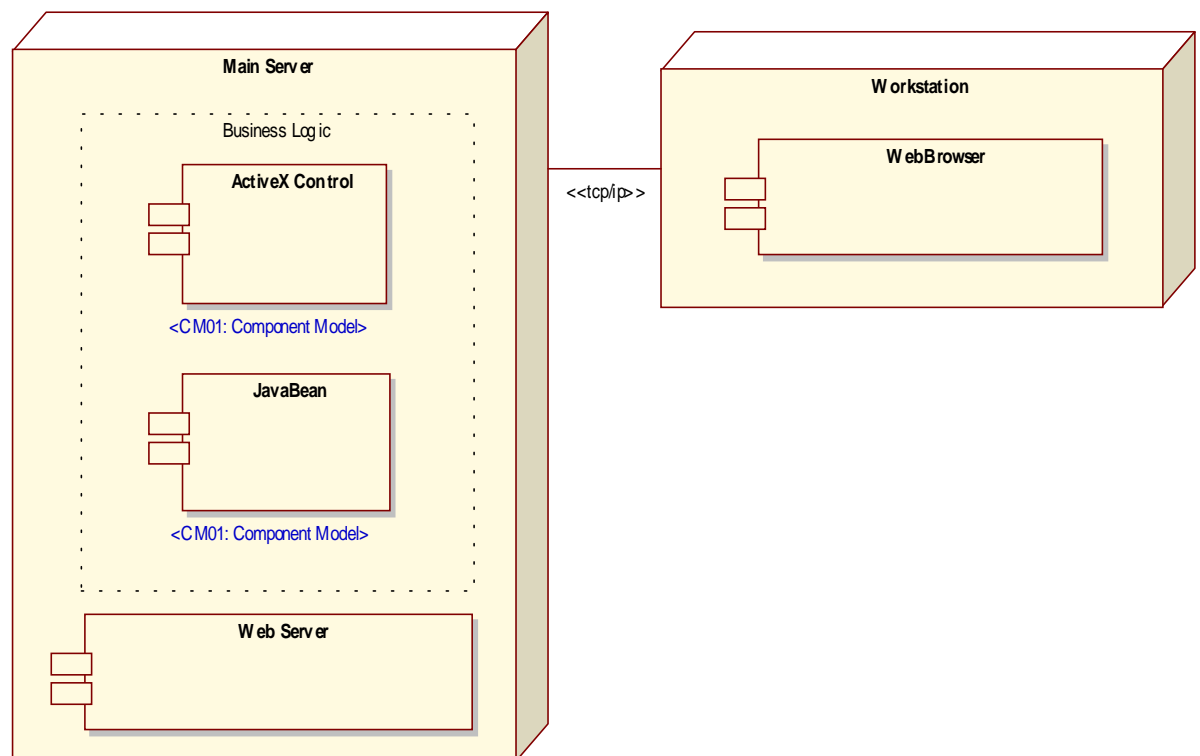
## Interfaces

Components may also expose interfaces. These are the visible entry points or services that a component is advertising and making available to other software components and classes. Typically a component is made up of many internal classes and packages of classes. It may even be assembled from a collection of smaller components.



## Components and Nodes

A deployment diagram illustrates the physical deployment of the system into a production (or test) environment. It shows where components will be located, on what servers, machines or hardware. It may illustrate network links, LAN bandwidth & etc.



## Requirements

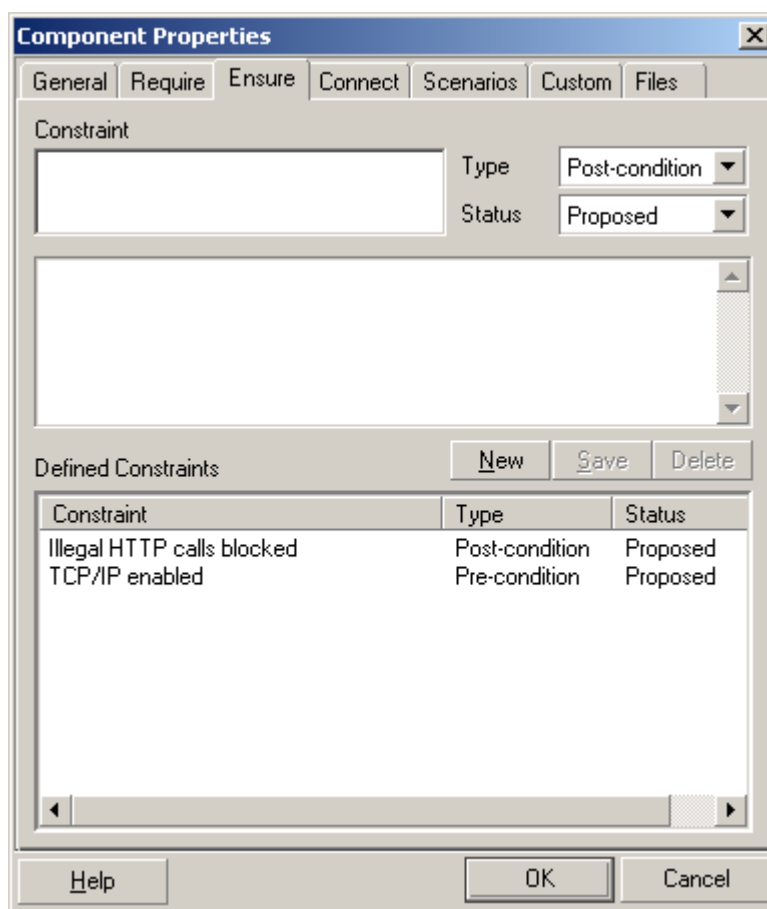
Components may have requirements attached to indicate their contractual obligations - that is, what service they will provide in the model. Requirements help document the functional behaviour of software elements.

The screenshot shows the 'Component Properties' dialog box with the 'Require' tab selected. The 'Requirement' field is empty, and the 'Type' dropdown is set to 'Functional'. The 'Status' is 'Proposed', 'Difficulty' is 'Medium', 'Priority' is 'Medium', and 'Last Update' is '29-Oct-2000'. Below these fields is a scrollable area for 'Defined Requirements' containing a table with two rows of requirements.

Requirement	Type
Block non-validated HTTP traffic	Functional
Pass secure HTTP requests	Functional

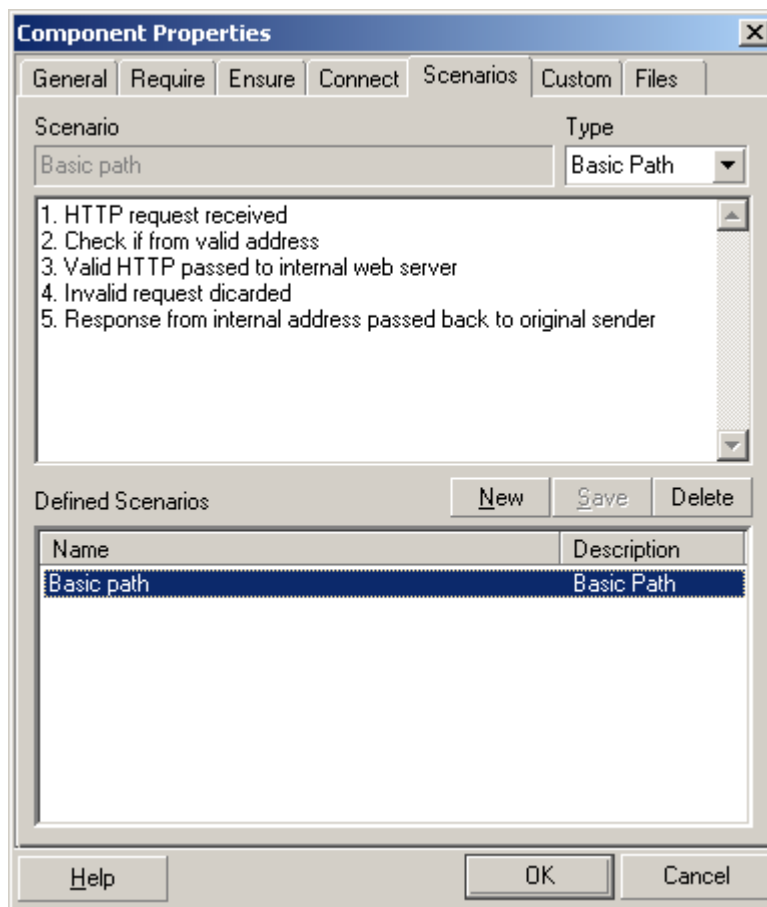
## Constraints

Components may have constraints attached which indicate the environment in which they operate. Pre-conditions specify what must be true before a component can perform some function; post-conditions indicate what will be true after a component has done some work and Invariants specify what must remain true for the duration of the components lifetime.



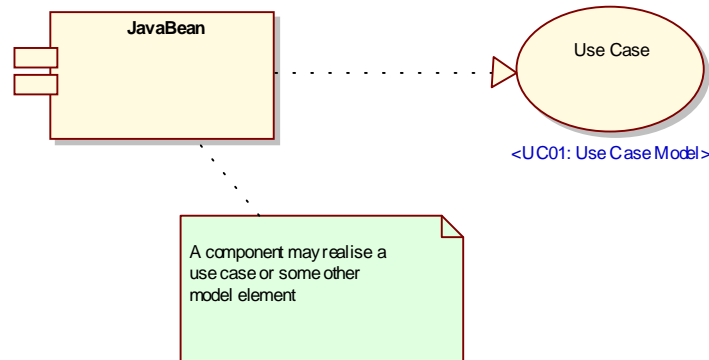
## Scenarios

Scenarios are textual/procedural descriptions of an objects actions over time and describe the way in which a component works. Multiple scenarios may be created to describe the basic path (a perfect run through) as well as exceptions, errors and other conditions.



## Traceability

You may indicate traceability through realisation links. A component may implement another model element (eg. a use case) or a component may be implemented by another element (eg. a package of classes). By providing realisation links to and from components you can map the dependencies amongst model elements and the traceability from the initial requirements to the final implementation.

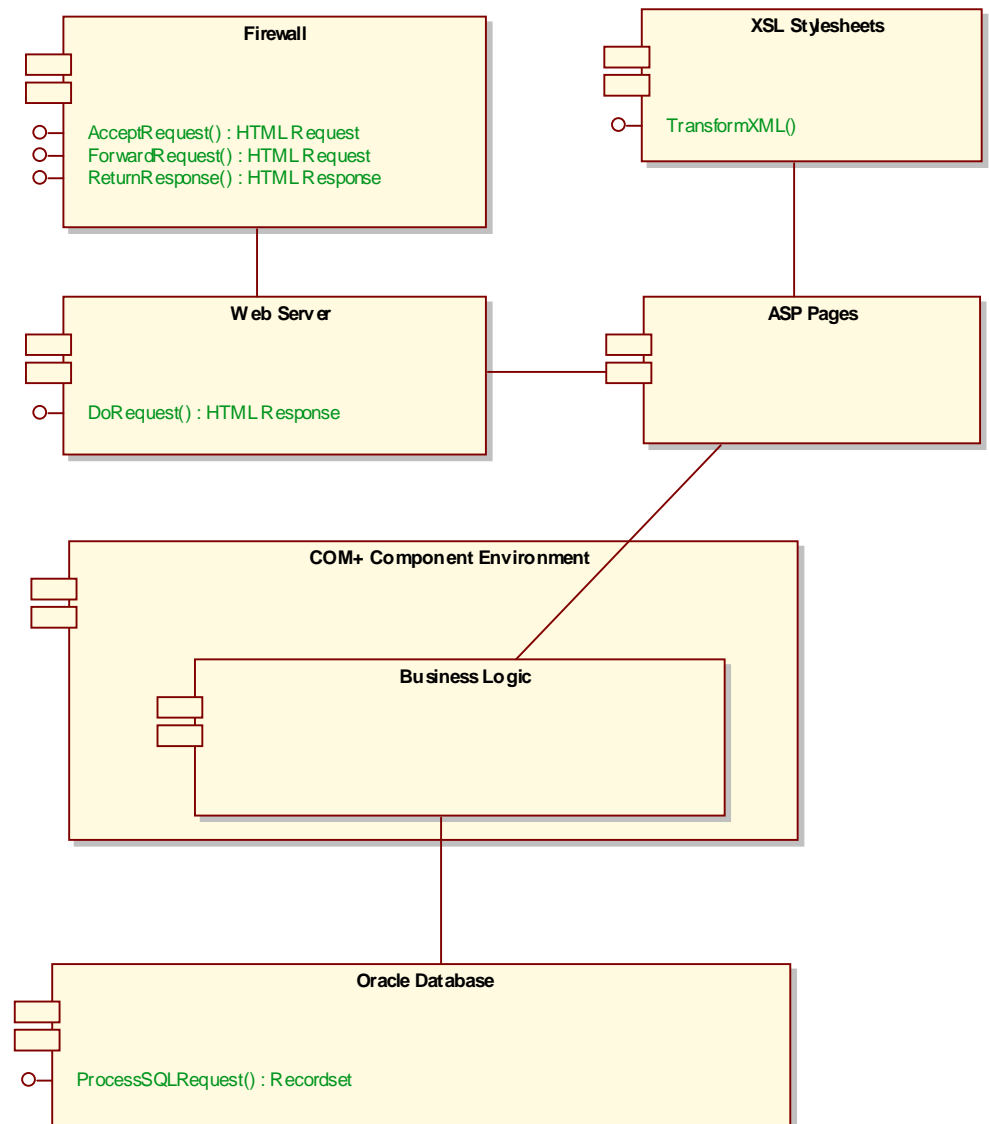


## An Example

The following example shows how components may be linked to provide a conceptual/logical view of a systems construction. This example is concerned with the server and security elements of an on-line book store. It includes such elements as the web server, firewall, ASP pages & etc.

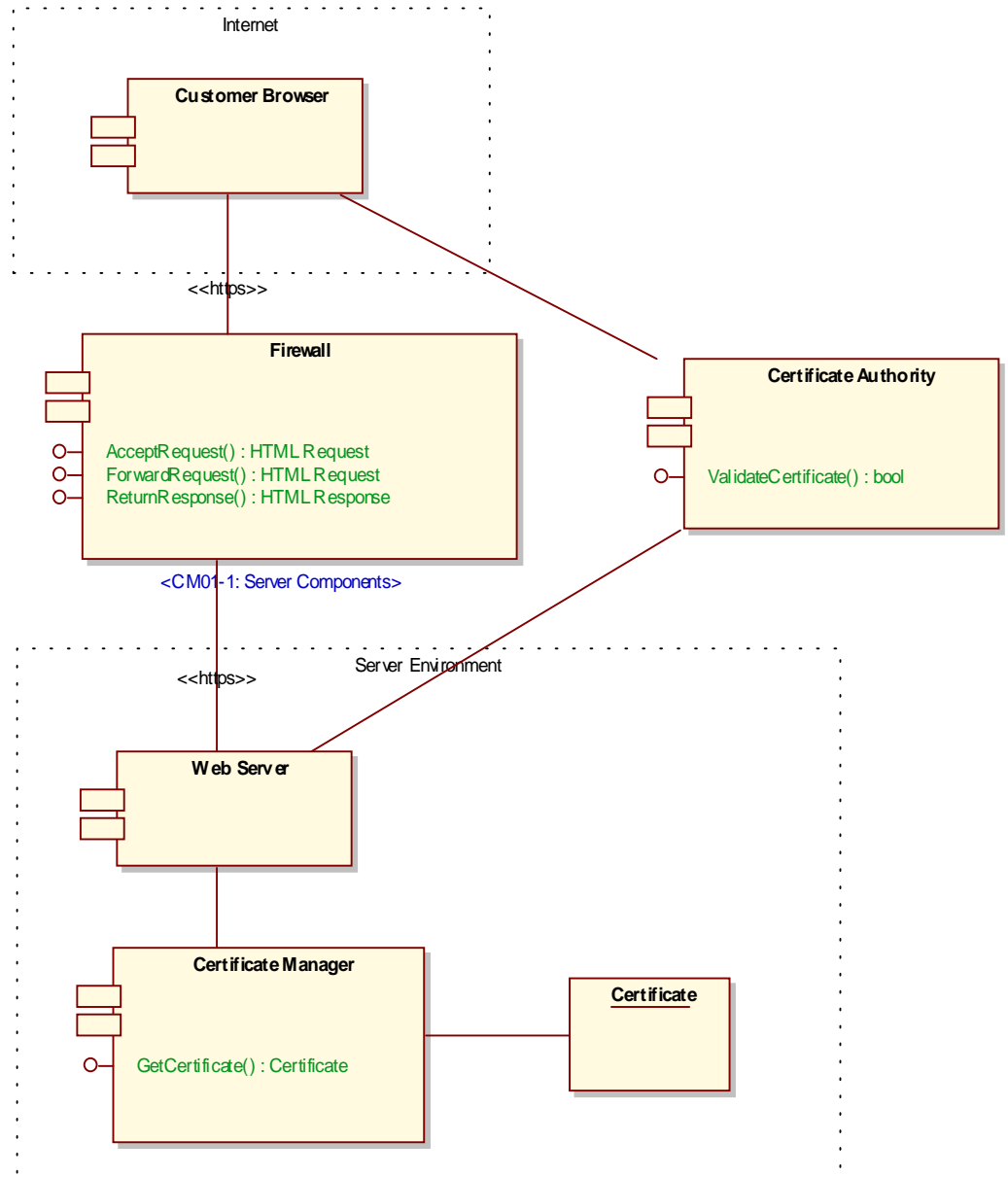
## Server Components

This diagram illustrates the layout of the main server side components that will require building for an on-line book store. These components are a mixture of custom built and purchased items which will be assembled to provide the required functionality.



## Security Components

The security components diagram shows how security software such as the Certificate Authority, Browser, Web server and other model elements work together to assure security provisions in the proposed system.



## Recommended Reading

Sinan Si Alhir, *UML in a NutShel*.

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

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

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

Geri Scheider, Jason P. Winters, *Applying Components*

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