SPARX SYSTEMS

UML TUTORIALS

THE COMPONENT MODEL

ENTERPRISE ARCHITECT

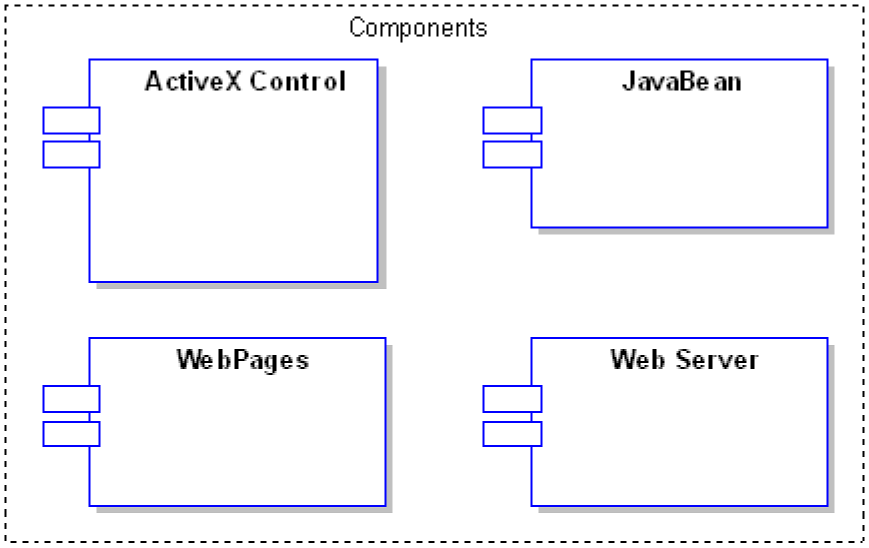
www.sparxsystems.com.au

The Component Model

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.

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:

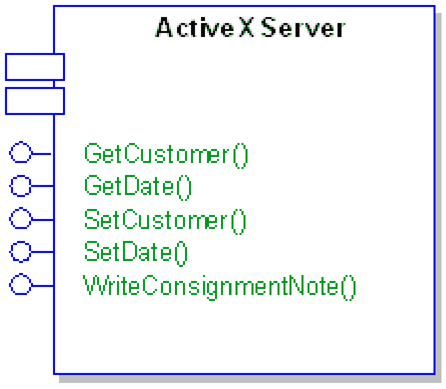


The Component Diagram

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

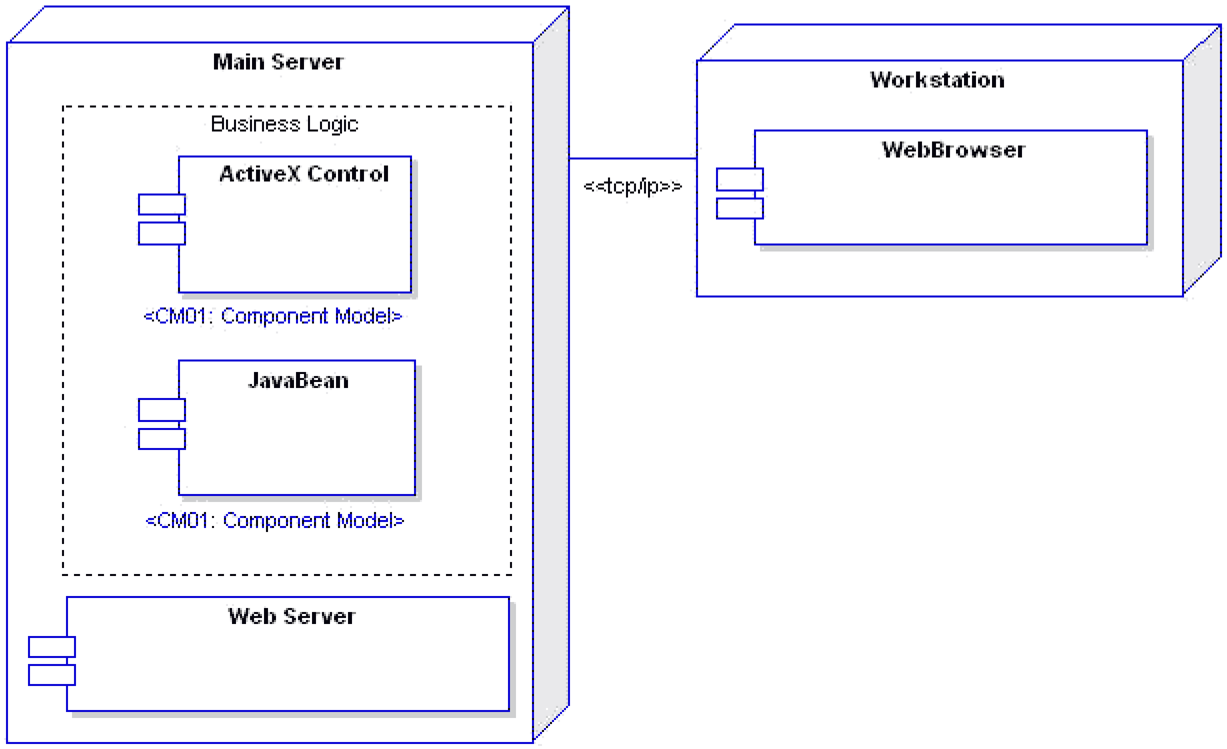
Interfaces

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.

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

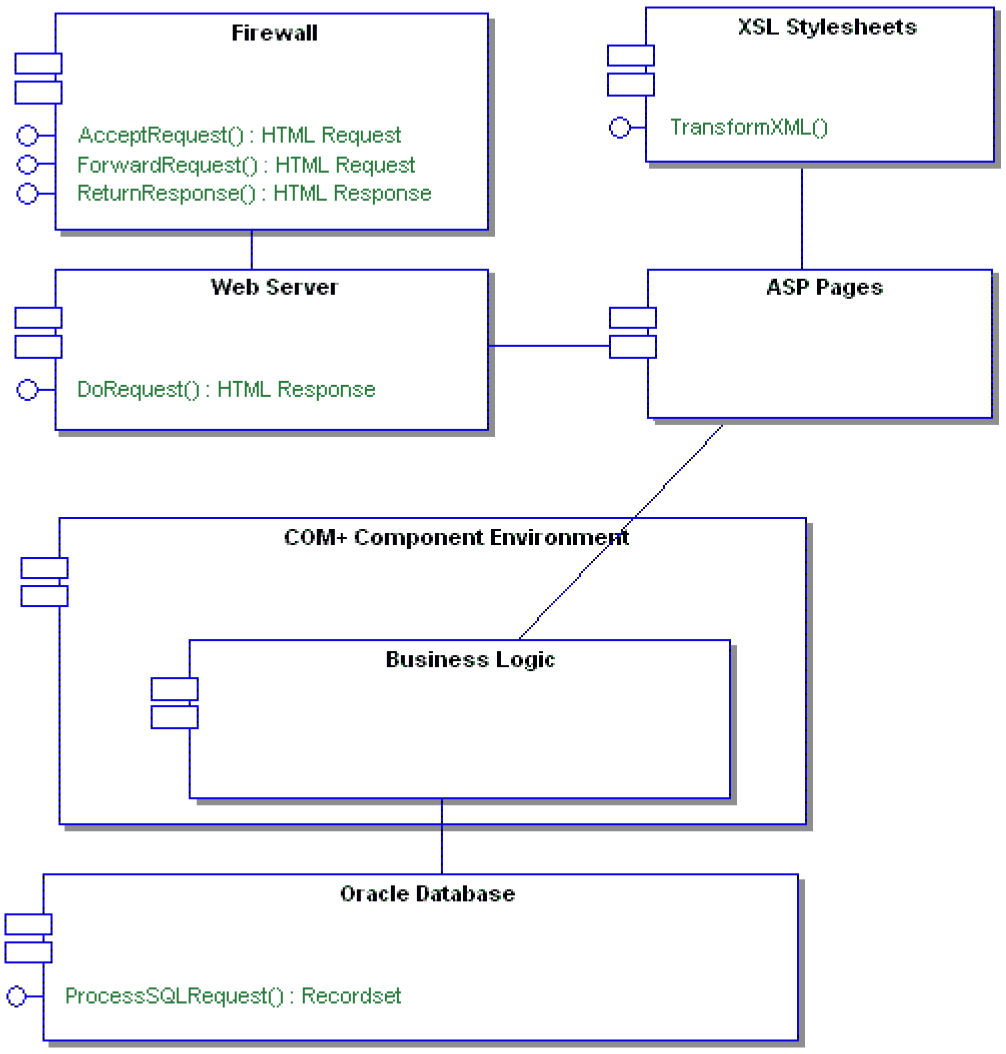
You may indicate traceability through realisation links. A component may implement another model element (eg. a use case) or a component may be implemented by another element (eg. a package of classes). By providing realisation links to and from components you can map the dependencies amongst model elements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

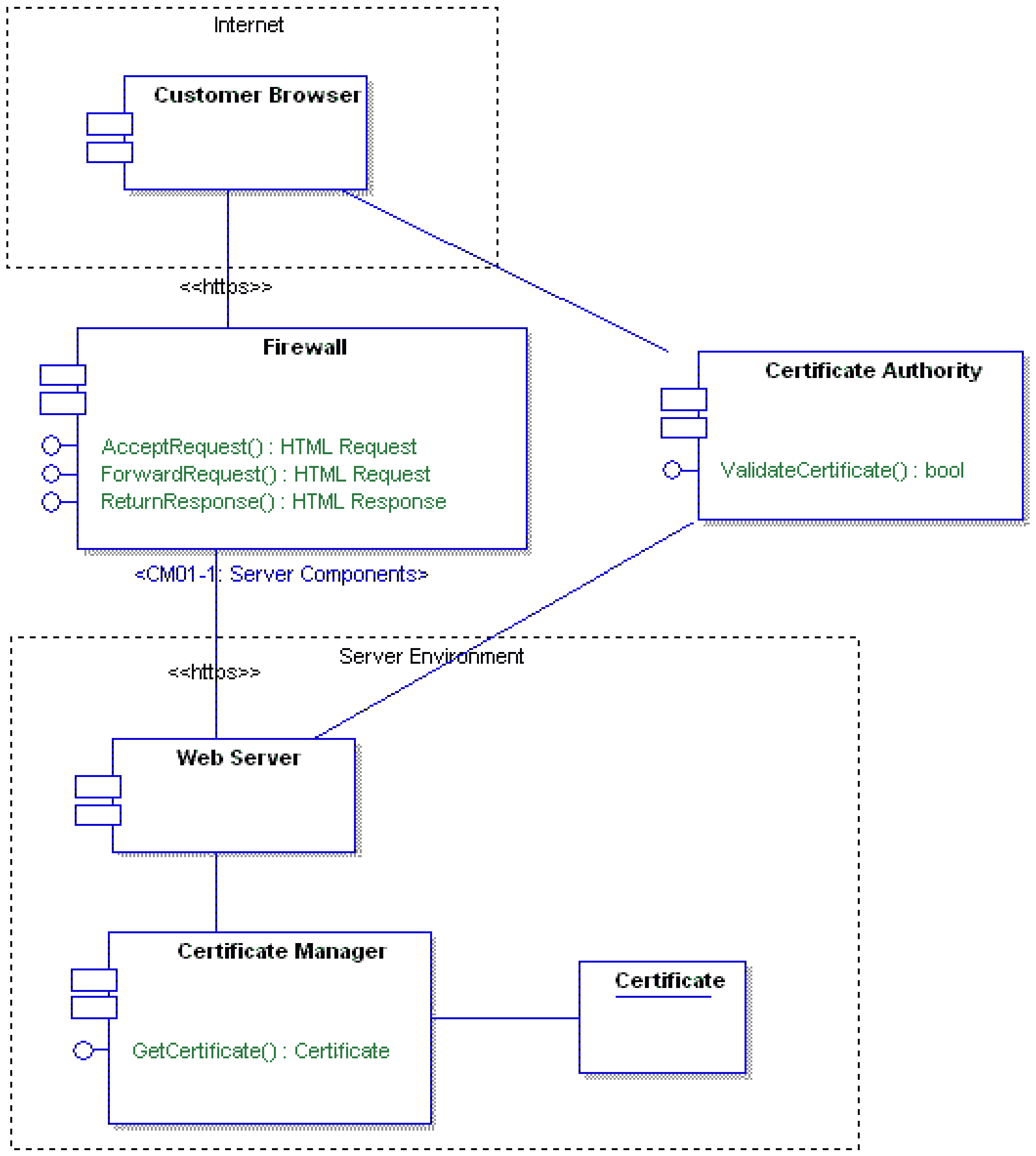
Server Components

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.


Firewall
AcceptRequest() : HTML Request
ForwardRequest() : HTML Request
ReturnResponse() : HTML Response
XSL Stylesheets
TransformXML()
Web Server
DoRequest() : HTML Response
ASP Pages
COM+ Component Environment
Business Logic
Oracle Database
ProcessSQLRequest() : Recordset

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.


Internet
Customer Browser
<<https>>
Firewall
AcceptRequest() : HTML Request
ForwardRequest() : HTML Request
ReturnResponse() : HTML Response
Certificate Authority
ValidateCertificate() : bool
<CM01-1: Server Components>
Server Environment
<<https>>
Web Server
Certificate Manager
GetCertificate() : Certificate
Certificate