Modelio
Practical Guides

Enterprise Architecture: Practical Guide to Logical Architecture

Author: Modeliosoft Consulting Team

Version: 1.0

Modeliosoft

21 avenue Victor Hugo

75016 Paris

www.modeliosoft.com

Introduction to Practical Guides

This set of Practical Guides is the result of hands-on experience gained by Modeliosoft consultants. Each guide is designed to facilitate model construction and to help you get the most out of the Modelio tool in a given context. The practical guides are deliberately short, since the aim is to provide essential practical information in just a few pages. The Modeliosoft consulting team is at your service to help with enterprise architecture definition, business process and software architecture modeling, SOA, and to provide any other assistance you may need in your IT projects.

Modeliosoft is pleased to provide a consulting/tool package. Find out more at www.modeliosoft.com.

At www.modeliosoft.com, you can download the Modelio Free Edition tool, a user-friendly and unlimited tool for UML modeling and business modeling (Enterprise Architecture, BPM, SOA logical architecture and software architecture), completely free of charge.

At www.modeliosoft.com, you can also evaluate and purchase Modelio Enterprise Edition, and discover the full functional richness of this tool: teamwork support, goal analysis, dictionary definition, requirements analysis, code generation, documentation generation throughout the entire project lifecycle, and so on.

The Practical Guides currently available are as follows:

- Practical Use Case Guide
- Practical Business Process Guide
- Enterprise Architecture: Practical Guide to Logical Architecture
- Practical Company Organization Modeling Guide

Other practical guides will be available soon. Please check our website for details.

What is information system logical architecture?

Logical architecture addresses the information system seen macroscopically, by focusing on its main components, their interconnections and the flows exchanged, and by structuring them by group into larger-scale modules.

One activity in the field of enterprise architecture is organizing the progressive and continual transformation of the information system, in the aim of simplifying it, optimizing its added value and making it more reactive and more flexible with regard to the company's strategic evolutions, based on the technological opportunities in the marketplace.

The information system map provides an overview of the information system's components. This will result in a consistent, stable and modular context to which different participants can refer in order to make all decisions regarding investment in the information system.

Through IS maps, existing application cartography is represented, as well as the target, in other words, the architecture expected after the programmed evolution of the information system.

Furthermore, logical architecture supported by Modelio SOA Solution enables the representation of ideal architecture, based on the architectural principles of service oriented architectures (SOA).

In this way, we can represent what already exists, by cataloguing application components that do not generally follow any particular overall architectural form, and we can model the target using an SOA approach or adopting a hybrid approach based on this approach.

When and how to articulate a logical architecture model

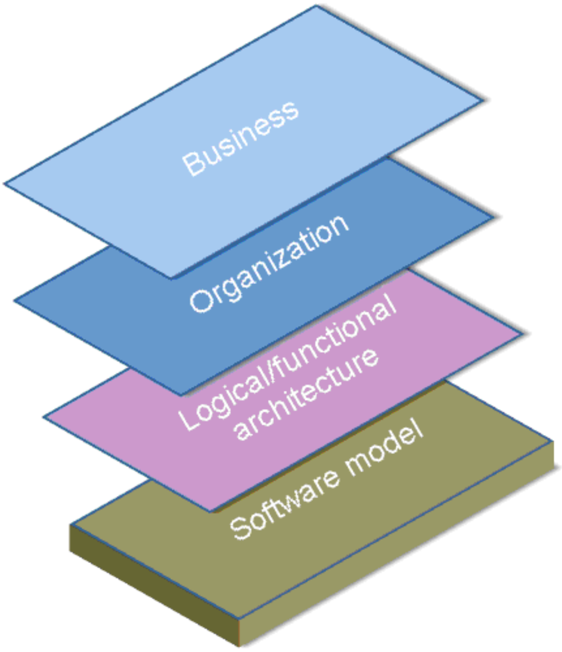


Figure 1 – Different models for the company and its information system

Figure 1 shows the positioning of the logical and functional architecture model with regard to the other models and work that surround an information system. Each of these models has a dedicated guide (see the Practical Organization Model Guide). Functional or logical architecture determines the components of the information system that are necessary to the functioning of the company. The logical/functional architecture model is established when the aim is to understand, document, adapt or improve the organization of the information system. Typically, it is used to build a common understanding of the information system through a general map, and to define the target for information system evolutions through the representation of future desired architecture. Ideal logical architecture is independent of technology. However, technical indications can be added to define the technologies used or retained for each component.

An ideal and theoretical scenario determines the following sequence:

1. Build the business model (for practical advice, see www.praxeme.org – semantic aspect guides).
2. Build the organization model (see the Practical Organization Model Guide).
3. Map and urbanize (in other words, organize and make coherent) the information system.
4. Build the logical architecture of the target.
5. Realize, acquire and assemble the application components.

Steps 3 and 4 can be undertaken simultaneously, even if the earlier models are absent.

Finally, the software realization stages are reached, where IT services will implement UML models focused on the information system and its realization, in accordance with the target logical architecture.

Best practices

Getting started with Modelio SOA Solution

Modelio SOA Solution is used to model urbanization and logical architecture. In the explorer, select the "Create Logical Architecture" button to create the root of the organization model. For this element, the palette contains the organization elements that can be created.

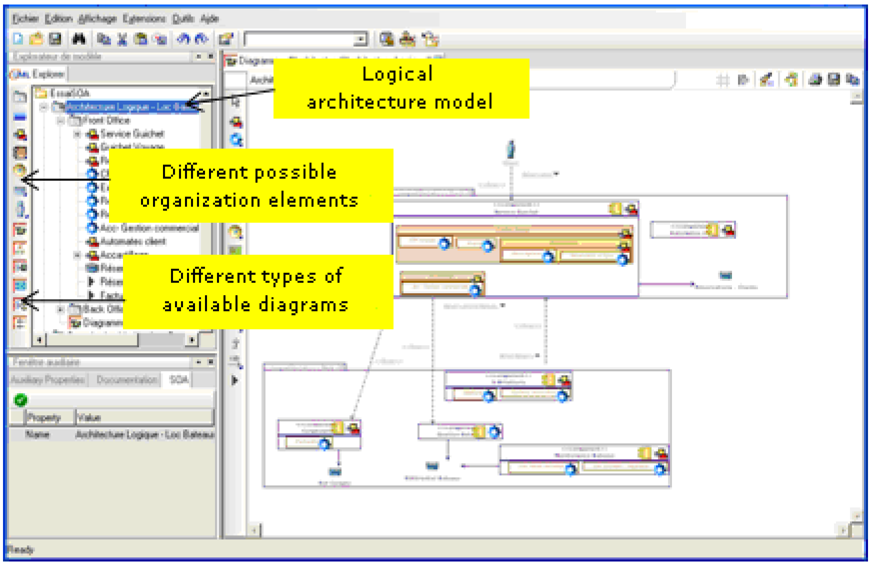

Fichier Edition Affichage Extensions Outils Aide
Explorateur de modèle
EssaiSOA
Architecture Logique - Loc Bateau
Front Office
Service Guichet
Guichet Voyage
Acc- Gestion commercial
Automates client
Back Office
Fenêtre auxiliaire
Auxiliary Properties
Documentation
SOA
Property
Value
Name
Architecture Logique - Loc Bateau
Ready
Logical architecture model
Different possible organization elements
Different types of available diagrams

Figure 2 – Starting the organization model with Modelio SOA Solution

We recommend that three different logical models (typically three roots in one Modelio project) be built to realize:

- the cartography of what already exists
- the urbanization of the target
- the logical architecture

The component composition and assembly technique already presented enables different occurrences (instances) of an element to be presented (component, quarters, application, …) in different models. The principle is similar to that shown in Figure 7. In this way, the target can use existing elements without redefining them, the logical architecture can break down identified applications in the cartography, and so on.

Building an information system map

Building an IS Map consists first of studying the different functional sectors of a company (production, administration, sales, …) to produce a cartography, and then studying the information system in the same way.

This type of approach starts with the inventory and capitalization of all available information on the company's information system (databases, applications, services, …) with regard to its functions, in order to rationalize and capitalize on the company's informational assets.

The aim of the IS map approach is therefore to end up with an information system structure that will enable the performance and progressiveness of the system to be improved. In this way, the company has the possibility of further developing its information system in full knowledge of the facts.

This approach uses different composition granularity levels, such as systems, sub-systems and applications:

- Systems, which are major functional domains supported by the information system
- Sub-systems, which break down large systems into coherent parts
- Applications, which correspond to autonomous elements providing well-identified services. Most of the time, these are already identified.

In a more general approach used to transit towards SOA architecture, Modelio SOA Solution supports these notions through systems (systems or sub-systems), applications and service components at a finer level of granularity. Component embedding (for example, the "Agency Reservation" application in the "Reservations" system, "Reservations" in the "Travel Counter" system, "Travel Counter" in the "Counter Service" system) is carried out by deploying an occurrence of each component in a higher-level component. In Modelio, this functions by creating an instance, according to the deployment principle described below).

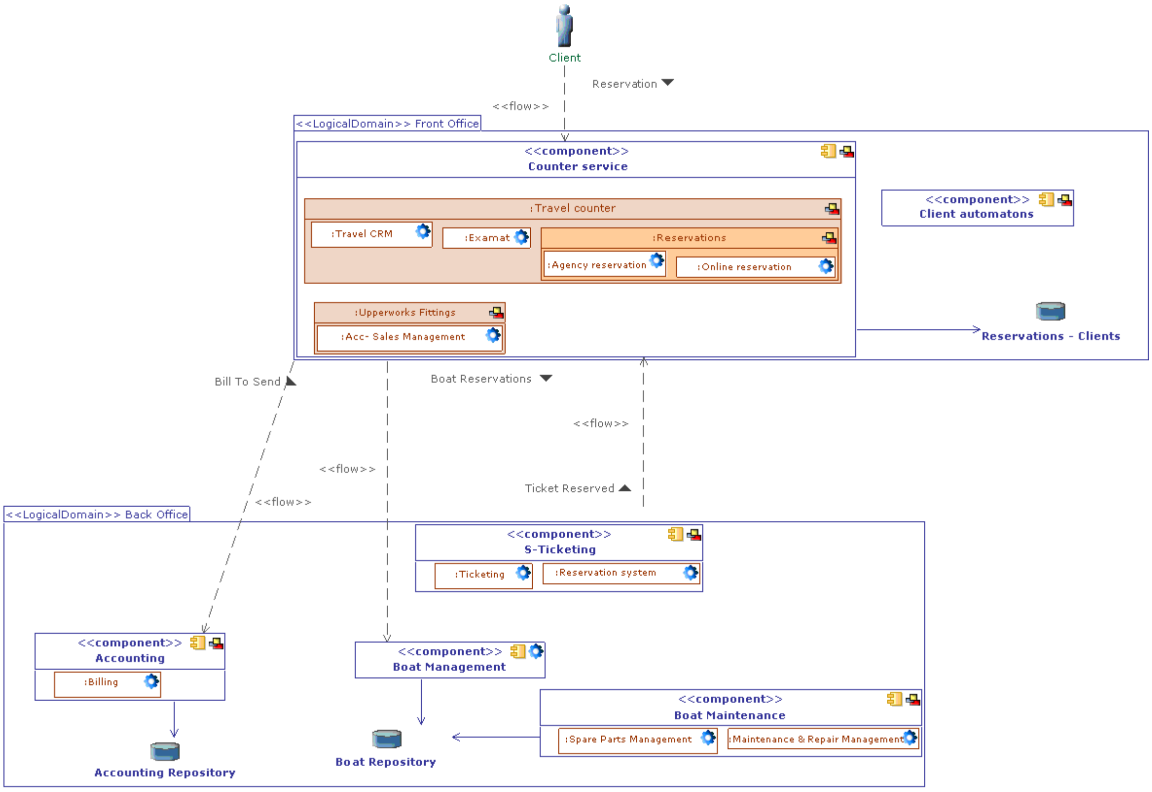


Figure 3 - Example of urbanization with Modelio SOA Solution (Modelio architecture diagram)

Logical architecture

Logical architecture plays a central role: a key tool in the construction and maintenance of the system, veritable pivot around which the business and its software translation take shape, logical architecture constitutes the reference for project organization, technical construction and progression planning. However, logical architecture is not an abstract model, nor a functional cartography of a remote target. In more pragmatic terms, logical architecture is the description of the components of the system and their relationships.

Logical architecture describes the system at a more detailed level than urbanization. It breaks down applications in order to end up with "service components".

Architecture will organize system components into layers, based on a typology linked to their level of stability. It will use the notion of service to guarantee the autonomy and interchangeability of components. There is a general consensus today for building system architectures from well established service typology organized into logical layers. Schematically, processes are based on a set of lower-level services and data access services.

Logical layers of increasing stability establish the basic dependency rule: a component cannot use a higher-level-layer component (for example, an Entity component must not use a Function or Process component).

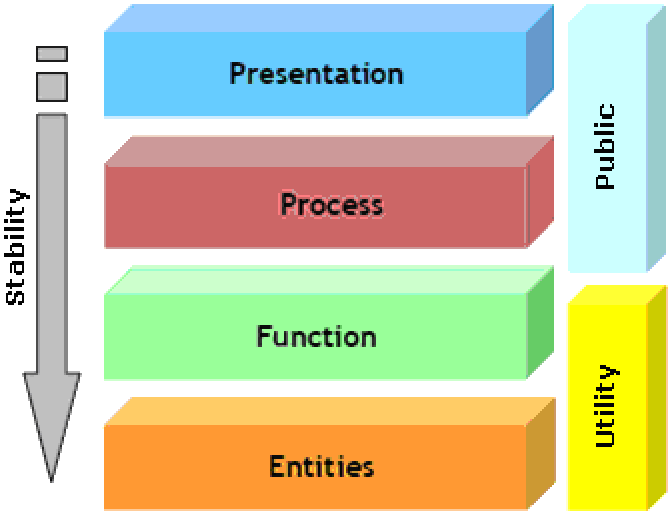


Figure 4 – Model in logical layers

Each type of component plays a specific role:

- "*Presentation*" components: Implementation of dialog with the user: GUI, user session management (this is not a service component in the strict sense of the term).
- "*Process*" components: Support of complete business processes (orchestration role); based notably on "Function" and "Entity" type components.
- "*Function*" components: Service composition. Functional adaptations or localized processing.
- "*Entity*" components: Access service to persistent data (CRUD12), databases and repositories.
- "*Utility*" components: Providers of infrastructure or transversal services (mailbox, dashboard, electronic publishing, address book).
- "*Public*" components: Dedicated to services that can be accessed from outside the information system (B2B, partners).

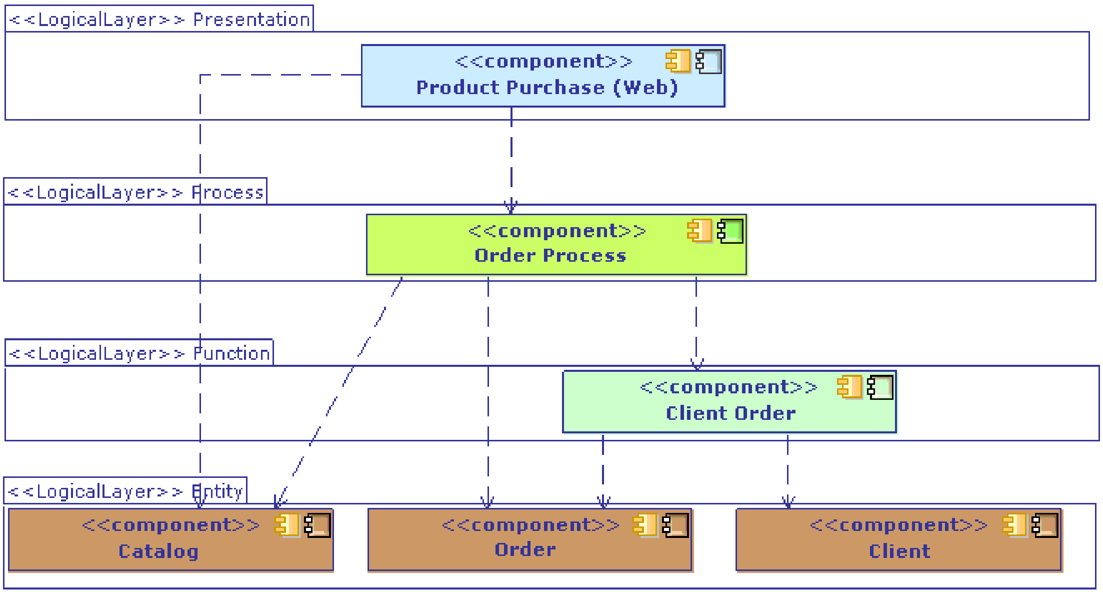

<<LogicalLayer>> Presentation
<<component>> Product Purchase (Web)
<<LogicalLayer>> Process
<<component>> Order Process
<<LogicalLayer>> Function
<<component>> Client Order
<<LogicalLayer>> Entity
<<component>> Catalog
<<component>> Order
<<component>> Client

Figure 5 – Service component architecture: example (architecture diagram)

The following table summarizes the criteria for the three main categories of component:

Type	Role	Type of participant	Granularity
🟩	Transversal business process. Service orchestration.	Service supplier and consumer	High granularity. Transversal by nature.
🟩	Handling process, service composition, adaptation	Service supplier and consumer	Medium granularity.
🟧	Access to a key business object	Service supplier	Fine granularity, focused on a key business object.

Rule: For every key business object, a corresponding Entity component must exist.

Rule: Process components come directly from the company's business processes.

Function components are gradually put in place through the successive consolidations of the system. They provide services that are close to the user vision, through the composition of Entity type services. (Example: "Client Contract" will be produced by aggregating "Contract" and "Client").

Entity type service components focus on a key business object of the system (for example Client, Contract, Order, …). Their role is to allow access to information relative to this business object, most often associated with a database. Write, read or request operations are typically found here. Any access to a key business object must go through the corresponding Entity component, which is unique.

Process type components automate a part of the business processes: these are handling processes. Process service operations are linked to process events: start, stop, or specific to the business.

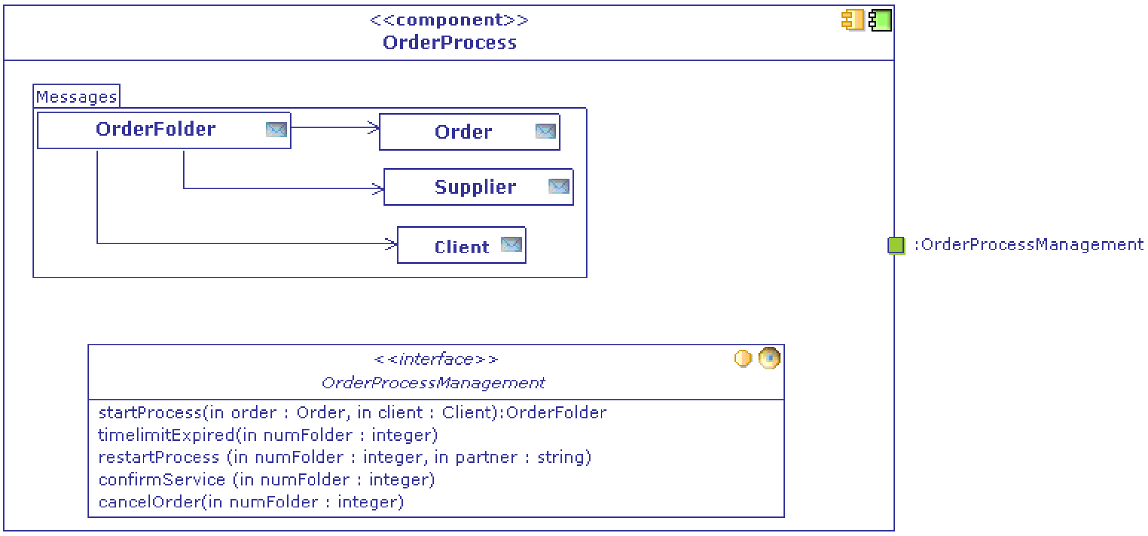

<<component>>
OrderProcess
Messages
OrderFolder
Order
Supplier
Client
:OrderProcessManagement
<<interface>>
OrderProcessManagement
startProcess(in order : Order, in client : Client):OrderFolder
timelimitExpired(in numFolder : integer)
restartProcess (in numFolder : integer, in partner : string)
confirmService (in numFolder : integer)
cancelOrder(in numFolder : integer)

Figure 6 – Process component: interface exchange data type (component diagram)

The approach used in identifying and constructing services will depend on the context of the company and the urbanization methods or models used, and will combine the following approaches:

- Business process approach: Determination of the business processes to be automated, based on the modeling of the organization and the processes (see the Practical Business Process Modeling Guide and the Practical Organization Modeling Guide).
- Data-oriented approach: the key business objects of the system.
- Application-oriented approach: restructuring of certain applications through service mutualization.

Elements essential to modeling the organization

The following table presents the main elements provided by Modelio SOA Solution for modeling the logical architecture and urbanization on an information system:

Icon	Name	Definition
🟩 🟩	Process component or process component instance	Supports a complete business process (orchestration role).
🟧 🟧	Entity component or Entity component instance	Logical representation of an autonomous business concept.
🟩 🟩	Function component or Function component instance	Intermediate component assembling different Entity or Function components or functions to provide a higher level service.
🟥 🟥	Utility component or utility component instance	Provider of infrastructure or transversal services.
🟨 🟨	Public component or Public component instance	Provider of public services.
🟦 🟦	Presentation component or Presentation component instance	Implementation of dialog and interface with the user.
🟨 🟨	System federation or system federation instance	Representation of systems or sub-systems, of their constitution in sub-systems/components and of their assembly.

⚙ ⚙	Application, Application instance	Federation of a set of components competing to provide services dedicated to a business line or a specific use of the system.
⛁ ⛁	"Database" component or component instance	Represents a shared data repository. Used for existing applications (cartography) where service components are not divided up. Service architecture does not present this type of component, since data management is taken care of by "Entity" components.
✉	Message	Messages or "exchange data types" represent the structure of flows exchanged through service operations.
● o {}	Service, Service operation	Functionality provided by the information system and made public, available and "invokable" by an interface (contract) so as to be mutualized or orchestrated.
🟩	Provided service	Access point to a service provided by a component. This access point is typed by the provided service.
🟦	Required service	Access point to a service required by a component. Provided and required services are linked by connectors.
🗀 🗀 UML 🗀	Package, Implementation Package, datamodel Package	These packages breakdown components or systems. The implementation package is used to express the implementation of a component in UML. The Package structures the elements of the logical model (typically interfaces or messages) inside components. Data models are limited to the static model to model data schemas (in general inside Systems).
🗀 L	Logical architecture unit	Very earliest structuring unit. Used to structure through functional domains, for example. Can be used to represent zones in urbanization.

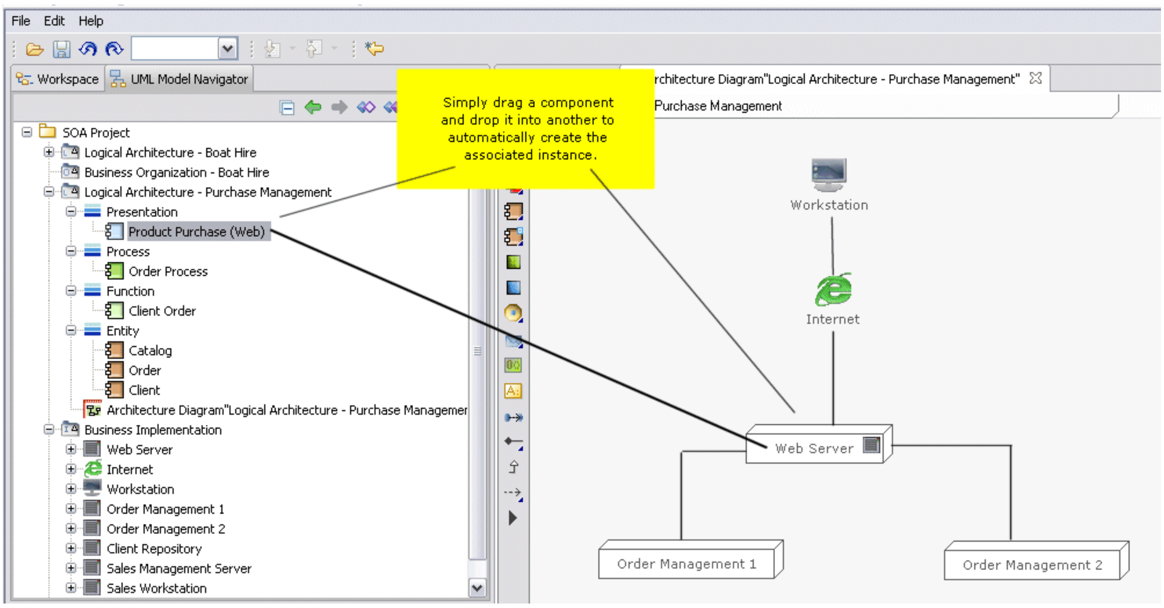
Material and geographical view – deploying information system components

The logical model will be even more explicit and understandable if its material and geographical deployment is defined. The different high-level system components are positioned on the material on which they will be run (servers, workstations, and so on). These material elements can also be localized geographically by deploying material in geographical locations. This work consolidates the way in which the current and future applications are deployed, and helps validate the architecture and its impact.

For this model, select the model root in the explorer and create the business deployment or business implementation model.

Icon	Name	Definition
🗄	Server	Central computer used as an application or data server. Components, systems, applications or databases are deployed on the server.
🖥	Workstation	Workstation of a system user. Used as an access point to the system. Components can also be deployed on a workstation. We recommend against deploying databases on workstations.
⚙	Application instance, …	Used to deploy an occurrence of a logical model element (system, application, component, and so on) on a site. (Graphically create the instance in the deployment area (for example, a server), and then type the instance using the site).

The model is then built simply by defining the material architecture elements (servers, workstations), and linking them through associations representing essential communication channels. Logical components are then deployed on the material architecture by creating instances typed by these units (see Figure 7). In this way, a single logical component can be deployed several times on different material.



Simply drag a component and drop it into another to automatically create the associated instance.

Figure 7 – Deploying a logical component on a server

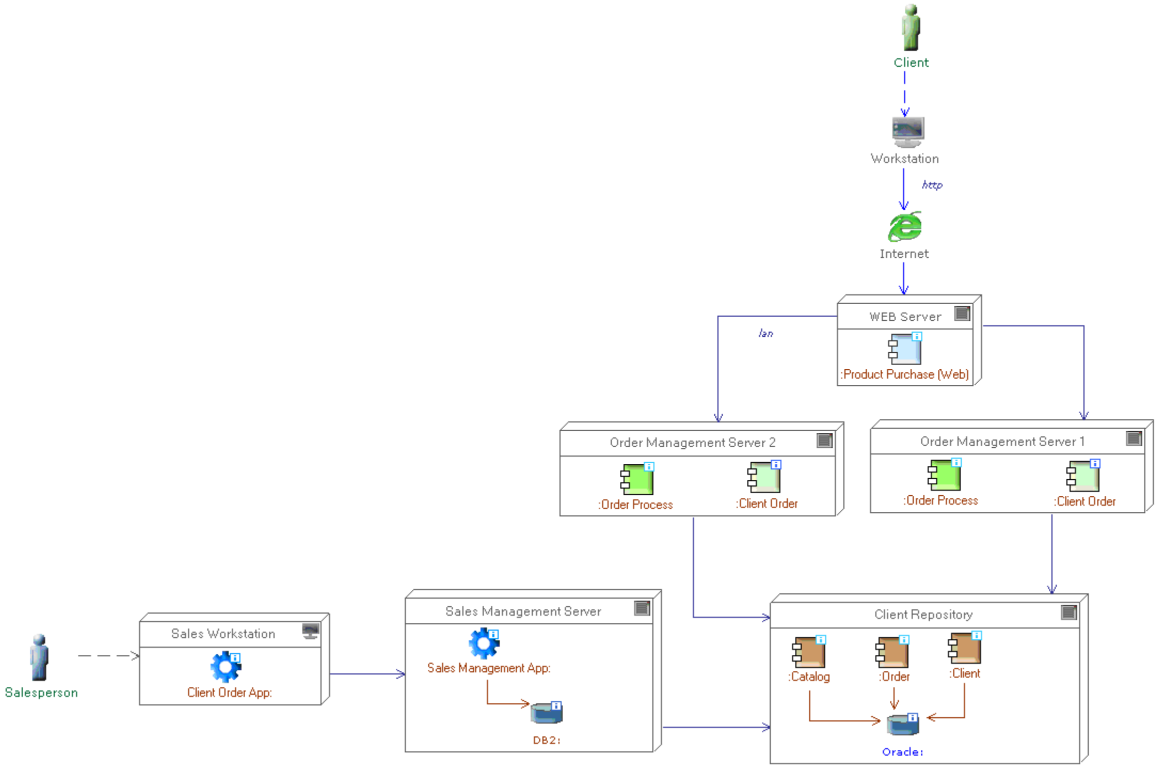


Figure 8 – Example of deploying logical architecture on material (physical diagram)