HOPEX IT Architecture

User Guide

HOPEX V2

INTRODUCTION TO HOPEX IT ARCHITECTURE

HOPEX IT Architecture is a comprehensive software product edited by **MEGA International** that allows you to represent and document your information system.

HOPEX IT Architecture enables description and analysis of the architecture of information systems.

- ✓ **Application mapping**: Application Architecture description.
  Description of application architecture offers a detailed view of information exchanges between applications, services, databases and organizational units.
  HOPEX IT Architecture also enables description of complex systems that involve resources other than software.
- ✓ **Application deployment**: technical infrastructure description.
  Information system technical infrastructure description enables monitoring of applications deployment on the different enterprise sites. The technical infrastructure takes account of the main hardware of your organization such as networks, servers, workstations, printers, firewalls and concentrators.
- ✓ **Information system management and upgrading**: city planning and service architecture description.
  IS service and city planning architectures are approaches that simplify IS upgrading by providing a frame of reference for planning of your systems and analysis of upgrading scenarios (with **HOPEX Planning**).
- ✓ **Representation of services provided by IT architecture**: description of internal structures of applications and application services.
  HOPEX System Oriented IT Architecture propose offers functions dedicated to representation and analysis of IT architectures using a service oriented approach.

HOPEX IT Architecture also offers an option that enables import of configuration elements from CMDB (Configuration Management DataBase) and their alignment with modeling objects described in **HOPEX IT Architecture**.

☛ *For more information, see the technical article "CMDB Import".*

PRESENTATION OF THIS GUIDE

This guide presents how to make best use of **HOPEX IT Architecture** to assure efficient management of your modeling projects.

> ☛ *Differences between **HOPEX Web Front-End** and **HOPEX Windows Front-End** are indicated where appropriate for each functionality.*

Guide Structure

The **HOPEX IT Architecture** guide contains the following chapters:

- "First Steps in HOPEX IT Architecture", page 9, presents how to connect and how to create main objects of a diagram.
- "Modeling Enterprise Architecture", page 21, explains how to perform more complex operations with **HOPEX IT Architecture**
- "Modeling Complex Infrastructures", page 57,presents how to model components and communication means of complex systems.
- "Application Deployment", page 87, describes how to deploy an application conforming to a required infrastructure.
- "Information System City Planning", page 95, explains how to describe Information Systems City Planning.
- "Service Oriented Architecture (SOA)", page 117, describes how to organize the service oriented architecture of your information system.
- "Detail of concepts used", page 127, presents the diagrams proposed and the concepts used in **HOPEX IT Architecture**.

Additional Resources

This guide is supplemented by:

- **HOPEX Common Features** guide describes the basic functions common to **HOPEX** products and solutions.

  > ☛ *It can be useful to consult this guide for a general presentation of the interface.*

- the **HOPEX Power Supervisor** administration guide.
- More advanced technical functions are described in the **HOPEX Power Studio** guide.

Conventions Used in the Guide

Styles and formatting

Remark on the preceding points.

Definition of terms used.

A tip that may simplify things.

Compatibility with previous versions.

Things you must not do.

Very important remark to avoid errors during an operation.

Commands are presented as seen here: **File > Open**.

Names of products and technical modules are presented in bold as seen here: **HOPEX**.

OVERVIEW OF HOPEX IT ARCHITECTURE

Reasons for Modeling your Information System Architecture

To face demands of competitivity, reactivity and efficiency, enterprises need constantly evolving information systems.

Technology now provides a wide variety of possibilities for building complex information system architectures. The increasing need for communication between applications, particularly using Internet, requires a more comprehensive and detailed consideration of how software resources are organized.

By representing this information in a detailed or general diagram, modeling helps to design and describe the architecture of information systems, facilitating any upgrades and enhancements.

There are many reasons for modeling an information system architecture. The modeler can describe the current system, focusing on the software architecture and how it is integrated within the organization, or on the technical infrastructure and IS system hardware. The modeler can also analyze which upgrades are required in an information system, whether this involves installing a new application, a new Internet site or a software package such as SAP, or optimizing a Client/Server architecture by finding how best to distribute data and processing.

The architect can also precisely represent the interactions between the applications of the enterprise and those of its partners, or between autonomous entities within the same enterprise as for example on installation of a Workflow engine.

In contrast to simple graphical representations or documents, modeling allows progressive building of a complete and consistent repository of information that is easy to maintain.

A Complex and Extensive Information System

To describe the architecture of a complex information system, the first step often consists of creating geographical overviews that show the primary sites of the company and how they communicate.

Depending on the level of detail, the modeling may show the entire information system in general, or may map more specific subsets.

For example, for an information system covering several continents, a world map would show the sites composing the system, organized by region or by country. Then specific regional maps would describe the local sites (headquarters, agencies at state level, etc.). For each map, the means used for sending data between the different sites are represented, as well as the site applications and hardware.

Creating Maps of the Applications and Databases

An information system is built in stages. Generally, enterprises have developed or purchased their applications when needed, when the technology or financial resources became available, etc.

Their accounting activities having been computerized, companies installed specialized tools focusing on their line of business. Then management tools arrived on the scene and we now see rapid growth in applications benefiting from Internet that combine the consultation and update technologies of geographically distant databases.

Now there is rapid growth in communication applications. All these applications developed gradually and sometimes independently of each other, but they now need to inter-communicate, share data, synchronize processing and have uniform interfaces.

Faced with these challenges, the enterprise that wants to control and manage the expansion of its information system has to have an inventory of its applications, of Internet/Extranet sites and the associated databases, and be able to locate where they are installed. It must also be able to define the functional and organizational scope of its applications (functions and users affected, information exchanged), and any interdependencies.

In this context, mapping of applications, Internet sites and databases becomes extremely beneficial to the enterprise.

The analysis of current hardware and software that mapping entails will reveal weak points in the information system: shortfalls, competing applications, data entered more than once, lack of uniformity. The enterprise can then decide how to fix these problems and what activities it can institute to improve organization and productivity.

The resulting maps are also a powerful tool when the information system is to be upgraded: they can be used to analyze the impact of a new application or software package on its future environment, formally specify EDI flows, or optimize a Client/ Server architecture.

City Planning of an Information System

Enterprise information systems develop with wide differences over time to suit projects. Under these conditions, it becomes increasingly difficult to integrate new applications into existing information systems and to handle technological evolutions. Information system city planning consists of defining the different components of an information system and defining their assembly terms so as to facilitate integration of these new applications.

In the same way as for a city, you can design the enterprise information system as for a city planning project and define a durable splitting of the information system into independent districts.

Defining the Functional Components of an Application

The advantages of breaking down an application into components can be felt throughout the life cycle of the application. This approach is essential in the

specification phase. The different requirements to be covered are then studied in sub-projects.

Defining the functional components maximizes the chance of successfully integrating a new application into the enterprise. During this crucial phase in the application life cycle, it is helpful to assign the different application functionalities to various org-units within the enterprise, in order to study their impact on the organization and set up any required training.

Also, representing application functional components facilitates their reuse.

Describing Interactions Between Applications

Interactions defined at an international level by organizations such as the OAG (Open Applications Group) or IFX (Interactive Financial Exchange), as well as the description of associated XML schemas (available with **HOPEX System Blueprint**) enable automation of exchanges between the applications of different enterprises. The use of interactions and standard messages enables these enterprises to communicate without prior knowledge of each other. This enables operation of market places, which are expanding rapidly. The same mechanisms can be used within an enterprise, as for example on installation of certain workflow engines, or to automate exchanges between an enterprise and its subsidiaries.

For more details, see "Describing Interactions", page 49.

Optimizing a Client/Server Architecture

Client/Server architectures are now found everywhere. This approach offers many possibilities in terms of reusing basic software components, portability and extensibility, so that information systems can be efficient and flexible no matter how complex the requirements.

However, not all application architectures are equivalent in terms of the information transported, time, risks, and costs.

Modeling helps answer these questions by defining the Client/Server architectures and breaking down the applications into client or server modules. Then the modeler can carefully consider where to locate the data and how to optimize its distribution and processing, based on a qualitative evaluation of the volume of information generated by the applications.

This means several different scenarios can be examined while still in the design phase, simulating alternative solutions of where databases are installed and replicated, or how application resources are shared.

Describing technical infrastructure

Enterprise information systems often combine legacy hardware and new equipment. There are often several generations of the same type of hardware coexisting in an information system, which only serves to increase complexity.

Modeling the technical aspects of the information system architecture often contributes significantly to the knowledge and management capabilities of system architects and managers.

The diagrams produced identify and represent existing hardware (servers, workstations, etc.) and the networks linking them, with descriptions of their characteristics. These diagrams indicate the machines hosting databases and applications.

FIRST STEPS IN HOPEX IT ARCHITECTURE

The aim of this chapter is to familiarize you with **HOPEX IT Architecture**: it introduces a fewfeatures of the software dedicated to diagrams use.

☛ *For more details on management of your desktop, and of diagrams and objects, see the **HOPEX Common Features** guide.*

The following points are covered here:

PREREQUISITES FOR USING HOPEX IT ARCHITECTURE

Certain options lead to a more optimized use of **HOPEX IT Architecture**.

Connecting to HOPEX IT Architecture

To connect to **HOPEX IT Architecture**, see HOPEX Common Features, "HOPEX Web Front-End Desktop".

Managing Message Options Messages

Certain functionalities described here are based on management functionalities for flows between the object of an application architecture based on the BPMN standard.

To access functionalities for managing flows in BPMN format :

1. Access the options window:
   - (***HOPEX MEGA Windows Front-End***) from the menu bar of your desktop, select **Tools > Options**
   - (***HOPEX MEGA Web Front-End***) from the **Miscellaneous** tool group of your **HOPEX IT Architecture** desktop, select **My Account > Options**.
2. In the options tree, select **Business Process and Architecture Modeling**:
3. Select the **Architecture Diagrams** option: **Use of BPMN Data Flows** option.

Use this option, selected by default, to apply BPMN formalism in your diagrams.

If, however, you want to continue using messages in your diagrams, an accounting option is available.

To activate the compatibility option:

1. Access the options window:
   - (***HOPEX MEGA Windows Front-End***) from the menu bar of your desktop, select **Tools > Options**
   - (***HOPEX MEGA Web Front-End***) from the **Miscellaneous** tool group of your **HOPEX IT Architecture** desktop, select **My Account > Options**.
2. In the options tree, select **Business Process and Architecture Modeling**:
3. Select the **Architecture Diagrams** option: **Use of Messages**.

If both of the options described here are selected, you can easily replace your messages with BPMN format flows.

USING DIAGRAM IN HOPEX IT ARCHITECTURE

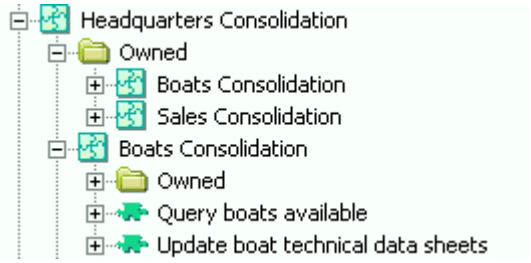
A project for describing the application architecture of an IT system consists in inventorying the existing applications and their interactions. The following points give some basics for the modeling of the application architecture.

Creating an application

HOPEX IT Architecture inventories applications in a navigation tree.

To create an application (out of a diagram):

1. Access the applications navigation tree:
   - (***HOPEX MEGA Windows Front-End***), from **HOPEX** menu bar , select **View** > **Navigation Windows** > **Main Objects**.
   - (***HOPEX MEGA Web Front-End***), from the navigation panes, click **Main Objects**.
2. Right-click the **Applications** folder and select **New > Application**. From the application created you can create its sub-applications, its services, as well as diagrams which will describe it.



For more details on applications, see "Information System Composition", page 22.

Creating a Diagram Describing an Application

HOPEX IT Architecture offers different diagrams to model application architecture of an information system. For more details on the list of diagrams available see "Detail of concepts used", page 127.

The creation of a diagram varies slightly depending on whether you are in MEGA Windows Front-End or MEGA Web Front-End.

HOPEX Windows Front-End

To create an application diagram in **HOPEX Windows Front-End**:

1. Access the application.

   For example from the application tree or from a diagram which describes the application.
2. Right-click the application and click **New > Diagram**.
3. Select the diagram type.

   For example "Application Architecture Diagram".
4. Leave the **Diagram Initialization** option selected (by default). This option makes objects available, automatically placed in the diagram created.
5. Click **Create**.
   The diagram created is automatically named by **HOPEX IT Architecture**. It describes the application which constitutes the described object. This application is automatically placed in the center of the diagram (only if you selected the **Diagram initialization** check box).

HOPEX Web Front-End

To create an application diagram in **HOPEX Web Front-End** :

1. Access the application.
2. Right-click the application and click **New** then the type of diagram.

   For example "Application Architecture Diagram".

   The diagram created is automatically named by **HOPEX IT Architecture**. It describes the application which constitutes the described object. This application is automatically placed in the center of the diagram.

Creating an "Overview of Applications" Diagram

The "Overview of Applications" diagram presents the main applications of the enterprise and their interactions.

In the same way you can create Overviews diagrams for sites and technical infrastructures.

HOPEX Windows Front-End

To create an "Overview of Applications" diagram in **HOPEX Windows Front-End**:

1. From menu bar , click **View** > **Navigation Windows** > **Main Objects**.
2. Right-click the **Overviews** folder and select **New > Diagram**.
3. In the dialog box that opens select "Overview of Applications".
4. Click **Create**.
   The diagram appears.

HOPEX Web Front-End

To create an "Overview of Applications" diagram in **HOPEX Web Front-End**:

1. Display navigation windows.

2. In the **Main Objects** navigation window, right-click the **Overviews** folder and select **New** > **Overview of Applications**.
   The diagram appears.

Creating org-units

> 🕮 *An org-unit represents a person or a group of persons that intervenes in the enterprise business processes or information system An org-unit can be internal or external to the enterprise. An internal org-unit is an organizational element of enterprise structure such as a management, department, or job function. It is defined at a level depending on the degree of detail to be provided on the organization (see org-unit type). Example: financial management, sales management, marketing department, account manager. An external org-unit is an external entity that exchanges flows with the enterprise. Example: customer, supplier, government office.*

To create an org-unit with HOPEX IT Architecture:

1. In the diagram objects toolbar, click **Org-unit**  .

   ☛ *When you click an icon in the diagram object bar (except the selection icon), its definition appears in the help window displayed by default at the bottom of the diagram.*
2. Click in the diagram where you want the org-unit to appear.
   The dialog box for creating an org-unit opens.
3. In the **Add a(n) Org-Unit** dialog box, enter the name of the org-unit you want to create.
4. Click **Create**.
   The org-unit appears in your diagram.
5. If necessary, resize the form so that the text fits.

An org-unit can also can be placed in a swimlane to improve diagram graphical presentation.

For more details on swimlanes, see "Using swimlanes" in the **HOPEX Common Features** guide.

Internal org-unit/external entity

During creation, org-units are considered as elements internal to the company.

To specify that an org-unit is not part of the company, you must modify the org-unit properties and enter the "External org-Unit" status.

To assign the "External Entity" characteristic to the org-unit:

1. Right-click the org-unit and select **Properties**.
2. From the **Characteristics** tab, click in the **Internal/External** field and select "External entity" value.
3. Click on **OK** to apply the update and close the properties dialog box.
   This characteristic is represented graphically and is automatically displayed in the diagram. When you close the properties dialog box of the "Boat Renting Agency" org-unit, its color changes.

   ☛ ***HOPEX IT Architecture** manages the characteristics of each object and these vary as a function of its type. These characteristics can*

*be modified in the **Characteristics** tab of the properties dialog box of each object.*

☛ *The characteristics of the different objects handled in **HOPEX IT Architecture** are described in "Concepts managed by HOPEX IT Architecture", page 139.*

Using flows

You can specify the content of message flows exchanged between the object of a diagram.

📖 *A message flow is information flowing within an enterprise or exchanged between the enterprise and its business environment. A message flow can carry a content.*

☛ *We recommend the use of BPMN flows. For more details on the options to be activated, see: "Managing Message Options Messages", page 10.*

Creating a Message Flow With Content

You can directly specify the content of *message flows* exchanged between the objects of a diagram directly at flow creation.

📖 *A message flow is information flowing within an enterprise or exchanged between the enterprise and its business environment. A message flow can carry a content.*

To create a message flow and its content:

1. In the diagram insert toolbar, click the **Message Flow** button arrow, option **Message Flow With Content**

2. Click the first object representing the start step, and, holding the mouse button down, draw a line to the object representing the next step.
   The **Creation of Message Flows With Content** dialog box opens.
3. In the **Content** drop-down list, select the content you wish to associate with the flow.
   The message flow is displayed with its content in the diagram.

   ☛ *You can associate several contents with the message flow. For more details, see "Defining Message Flow Content", page 14.*

Defining Message Flow Content

To define content of a message flow:

1. Right-click the message flow and select **Properties**.
   The properties window opens.
2. Select **Characteristics**.

3. Click the arrow at the right of the **Content** field and select **Connect Content**.
   The selection dialog box appears, with a list of contents proposed for the message flow.
4. Select the content name and click **OK**.

   ☛ *A content can be used by several message flows since it is not associated with a sender or recipient.*

   The name of the content appears in the diagram.

Creating a message

We recommend the use of BPMN flows, but the use of messages remains available with a compatibility option. For more details, see: "Managing Message Options Messages", page 10.

A message represents information flowing within an enterprise or exchanged between the enterprise and its business environment. Messages can be information flows such as orders or invoices. For convenience, financial and material flows such as payments or product deliveries are also represented by messages.

In a diagram, a *message* is described by:

- A route, which indicates its sender and its receiver.
- A content, which represents the objects or information that transit from message sender to receiver. This content can be reused in several messages.

There are two ways of creating messages. You can:

- create your messages while connecting them to the sender and recipient.
- create your messages first and then connect them to the sender and recipient.

Creating messages while connecting them

To create your messages while connecting them to the sender and recipient:

1. In the diagram objects toolbar, click **Message**  .
2. Select the sender (external org-unit, application or database) and drag the cursor to the recipient object before releasing the mouse button.
   The *Add Message* dialog box opens, in which you enter the name of the content used by the message.
3. Enter the message content name and click **OK**.

   ☛ *By default the message carries the same name as its content.*

   ☛ *When the message content already exists in the **HOPEX** repository , the **Content already exists** message is displayed at the bottom of the dialog box. Click on **OK** to use this content. Otherwise, modify the message name in the **Name** field and click **OK**, to create a new content for this message.*

Note that the message is automatically placed in the middle of the line connecting the sender and recipient.

At times you may need to move messages: for example, if two objects exchange more than one message, these may overlap and you will need to move one of the messages so that both are visible.

Creating messages first and then connecting them

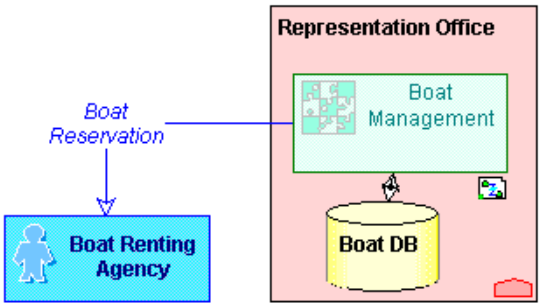
To create *messages*, and then connect them to the sender and recipient:

1. In the diagram objects toolbar, click **Message** .
2. Click in the diagram where you want to see it appear.
   The Add Message dialog box opens, in which you enter the name of the content used by the message.
3. Enter the message content name and click **OK**.

To create each link:

1. In the diagram objects toolbar, click **Link** .
2. Select the source object (sender or information flow) and drag the cursor to the target object (information flow or recipient).

Below the "Boat Reservation" message is sent by the "Representation Office" to the "Boat Renting Agency".



When an information flow (message) has several senders and recipients, you can create it using the first technique and then connect it to the other senders and/or recipients.

An information flow may have special characteristics that impact how often or when it is sent. In this case, a **Timer** that you connect to the corresponding information flow allows you to indicate this characteristic.

Managing Specialized Views

HOPEX IT Architecture allows you to restrict or extend the range of concepts that you want to feature in a diagram.

You can build very comprehensive diagrams by superimposing specialized layers as with tracings:

- layers that concern software-related objects (sites, applications, databases, etc.)
- layers that present org-units using the applications
- layers that complete representation with computer equipment used.

To select the appropriate display mode:

- In the diagram toolbar, select **Views and Details** .
  By default, the display only shows the drawing objects and the standard view of the current diagram (the basic concepts and the links between them)..

To extend the field of concepts used in a diagram:

- In the **Views and Details** dialog box, select the views that correspond to the required extensions.

To restrict the view:

- In the **Views and Details** dialog box, clear the view that corresponds to the specialized view.
  This allows you to avoid displaying objects you no longer need to see in the specialized view. The objects removed and their links with other objects remain memorized by **HOPEX IT Architecture**. They can be displayed later in the same location by reselecting the corresponding view.

  For example, to be able to represent the org-units of the travel agency and the working processes, as well as the server supporting the agency applications and databases, check the **Org-Units**, **Servers** and **Workstations** check boxes.

  The corresponding icons appear in the objects toolbar.

Improving the Layout of your Diagram

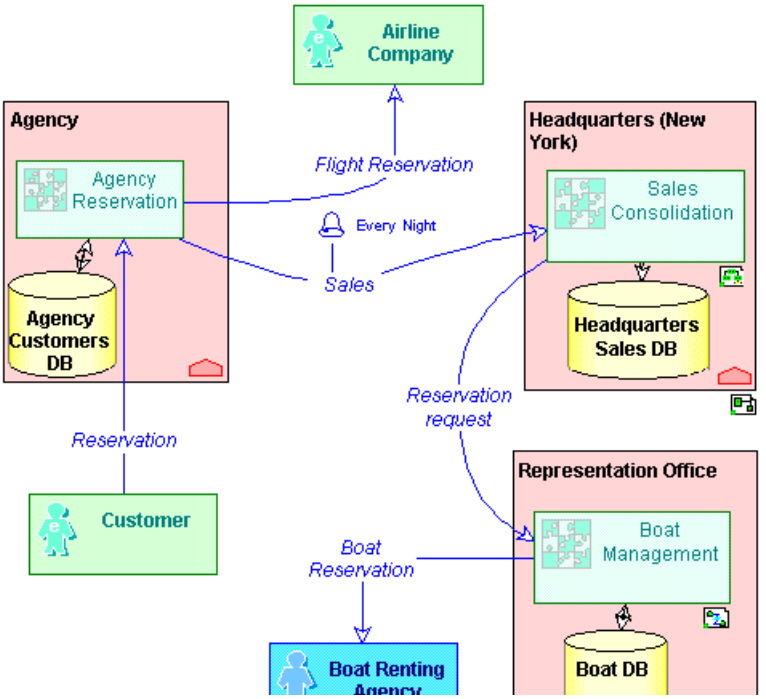
Having created all the objects and links, you can improve the layout of your diagram. You can, for example:

- move objects
- align objects (menu **Drawing** > **Align**)
- modify the shape of links
- Add joints to links using the <Ctrl> key so as to go around obstacles or in order to draw curves, etc.

☛ *For more information, see **HOPEX Common Features**.*

☛ *For more details, see "Handling Diagrams", page 65.*

Below, to improve readability of your diagram, the sites have been enlarged to accommodate all the objects they "contain".



2

MODELING ENTERPRISE ARCHITECTURE

This chapter describes concepts linked to modeling of application architecture and diagrams responding to various requirements of the information system architect.

The following points are covered here:

DESCRIBING AN APPLICATION ARCHITECTURE

Application architecture is the logical organization of applications and their exchanges, providing a summary representation of all or part of the information system.

Application mapping projects are based on inventory of existing application assets. Modeling of these assets within a repository provides a tool for controlling what exists and serves as a basis for implementation of development projects.

Information System Composition

The information system can be broken down according to two detail levels: application and application system.

Applications

An application is a set of software components constituting a coherent whole regarding deployment, functional coverage and IT techniques used.

The application is the management and deployment unit of a set of software components. An application can be deployed on one or several machines. An application responds to:

- business requirements

  Examples: billing, accounting, equipment management, load/capacity calculation.

- technical requirements

  Examples: specific communication interface, access control.

- transverse requirements

  Examples: electronic mail, directories, office system applications.

For the creation of applications, see "Creating an application", page 12.

Application System

An application system is an assembly of applications responding to a coherent set of functionalities, implemented by the applications making up the system.

Application systems can:

- comprise a suite of applications grouped for commercial reasons (integrated management software packages such as SAP, Oracle Applications, Siebel…) or
- correspond to a group of applications that have the same functional objectives (accounts and financial management system integrating all

accounting applications: general, suppliers, analyses, as well as financial and budgetary analysis modules, human resources management systems integrating salaries, time management, career management, etc.).

The application system and application can be the subject of specific developments (carried out internally or bought-in/sub-contracted) or they can be proprietary market products (software packages).

The logical organization and structure of application systems and applications, together with description of their exchanges, constitutes the foundations of the application architecture.

In **HOPEX IT Architecture**, we use the terms application and application system without distinction to name components of an application architecture.

Applications listing

Breakdown of an application system into sub-applications (or sub-systems) is considered from two main viewpoints:

- on the one hand we consider the software elements provided by the application system (system deployment involves deployment of its components, system development is dependent on development of its components, etc.).
- on the other hand we consider the application in terms of its operation. In this case we must also include the sub-applications used on execution of the application, but which are not supplied.

  We are talking here about applications used explicitly; the user application knows the application used and calls its services directly.

Exchanges between applications

See "Describing Interactions", page 49.

Managing Application Versions

The information system of an enterprise is in constant evolution; new applications may be integrated, but in most cases modifications to the IS consist of adapting existing applications. It is therefore important to be able to monitor evolutions of these applications and to identify differences between each version.

HOPEX IT Architecture enables management of application versions and more complex architectures:

- based on a modeling technique: variation, and
- by locking the application.

If you create a version of an application, the application is automatically locked.

To modify the application, you must create a new version or unlock the application.

☺ *To use variations (and consequently, versions), you must select user option **Business Process and Architecture Modeling** > **Activate Variations**.*

☛ *For more details on variations, see the **HOPEX Common Features** guide, "Handling Repository Objects", "Object Variations".*

DESCRIBING AN APPLICATION ENVIRONMENT

HOPEX IT Architecture allows you to refine your analysis of the information system. Such an analysis usually focuses only on a small part of the information system.

In this way you can create an environment diagram of a specific application, describing information flows exchanged, databases used, org-units and procedures concerned.

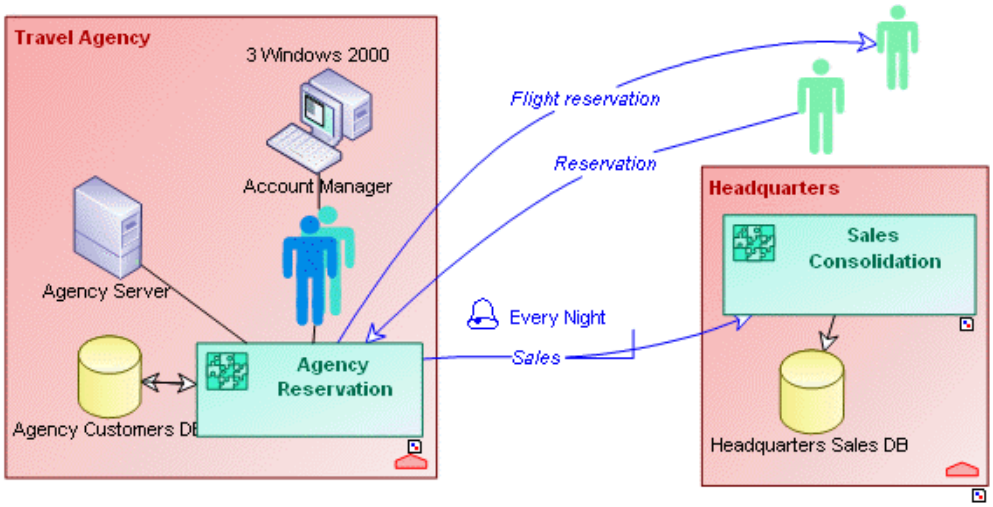
The environment diagram enables placing of the application in its use context. In it are defined the exchanges identified or expected (service contracts).

*Rather than use environment diagrams, **MEGA** recommends that you use internal application architecture diagrams to reference the applications. You must therefore create "system application" type parent applications (eg: Accounting Application System). This method is used to improve, in particular, the description of inter-applicative flows that will then be defined within an application system.*

The environment diagram is not used to describe breakdown into application modules or services (see application tree). Nor does it contain the applications and services necessary for its operation (see internal architecture diagram).

Its precise use framework enables proposal of suitable behaviors when introducing the application or service into the diagram. All applications added to the diagram are considered as "neighboring applications", and as a result there is no automatic link addition (breakdown or internal architecture participation link).

You can also add to the diagram technical infrastructure elements such as the servers hosting the applications and databases, and the network connecting the servers with agency workstations.



The icon that appears under certain objects indicates they are described by a diagram..

For more details on diagrams and described objects, see:

✓ "Detail of concepts used", page 127

Prerequisites to Using the Environment Diagrams of an Application

To access the environment diagrams of an application:

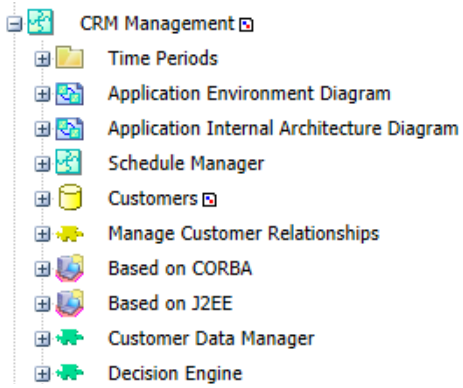
1. Access the options window:
   - (***HOPEX MEGA Windows Front-End***) from the menu bar of your desktop, select **Tools > Options**
   - (***HOPEX MEGA Web Front-End***) from the **Miscellaneous** tool group of your **HOPEX IT Architecture** desktop, select **My Account > Options**.
2. In the options tree, select **Compatibility > Diagrams**.
3. Select the **Creation Application for an Application Environment Diagram** option.

Opening an Application Environment Diagram

To open the environment diagram of an application:

1. In the **Main Objects** Navigation window, expand the **Applications** folder.
2. Expand the required application folder.

   Example: the "CRM Management" application.



3. Double-click **Application Environment Diagram**.

Creating an Application Environment Diagram

The procedure varies slightly depending on whether you are in MEGA Windows Front-End or MEGA Web Front-End.

HOPEX Windows Front-End

To create an application environment diagram from **HOPEX Windows Front-End**:

1. Access the application.

   For example from the application tree or from a diagram which describes the application.
2. Right-click the application and click **New > Diagram**.
3. Select diagram type "Application Environment Diagram".
4. Leave the **Diagram Initialization** option selected (by default). This option makes objects available, automatically placed in the diagram created.
5. Click **Create**.
   The diagram created is automatically named by **HOPEX IT Architecture**. It describes the application which constitutes the described object. This application is automatically placed in the center of the diagram (only if you selected the **Diagram initialization** check box).

HOPEX Web Front-End

To create an application environment diagram in **HOPEX Web Front-End**:

1. Access the application.
2. Right-click the application and select **New** > **Application Environment Diagram**.
   The diagram created is automatically named by **HOPEX IT Architecture**. It describes the application which constitutes the described object. This application is automatically placed in the center of the diagram.

You can access this diagram from any diagram containing the described application.

☛ *See the **HOPEX Common Features** guide for more details on creation, handling and adding objects in a diagram.*

☺ *If you are creating a new diagram based on an existing one, we recommend that you simply copy and paste the objects you need from the existing diagram into the new diagram., see "Copying Objects from Existing Diagrams", page 27.*

Copying Objects from Existing Diagrams

In a diagram corresponding to a partial view of the information system, you can reuse elements from a more general diagram useful in this new perspective. You can then create the additional objects that you need to obtain the level of detail required.

For example you can copy objects of an Overview created by IS Management and paste them in an Environment Diagram of one of the described applications.

Select and copy for example the sites, applications and databases you want and paste them in the target diagram.

This operation also allows you to recover previously defined links and formatting.

You can also use multiple selection of objects:

- Holding down the <Shift> key, click successively on the objects you need.

Reusing Objects

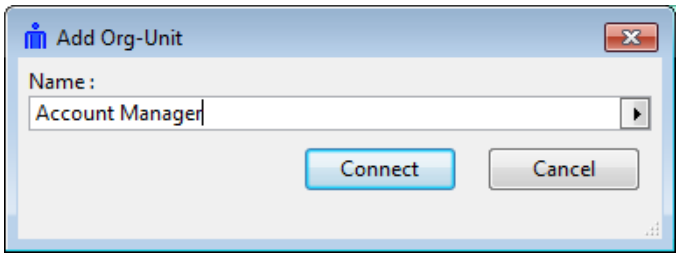
Certain objects already exist in the **HOPEX** repository.

To reuse in a diagram an org-unit already created:

1. Access the diagram.
2. In the insert toolbar, select **Org-Unit**  .
3. Click in the diagram where you want to see it appear.
   A creation dialog box opens.
4. Enter the name of the org-unit.
   The **Create** button changes to **Connect** as soon as you enter the last letter of the name of the org-unit already existing in the repository (example: the "Account Manager" org-unit). This change indicates that the org-unit (example: "Account Manager") is known to the system.

   ☛ *If the button still reads **Create**, this is because a typing error has prevented recognition of the name entered: an additional or missing letter for example.*

   The name is not case-sensitive.



5. Click **Add**.
   The org-unit now appears in the diagram with all its links existing in the repository..

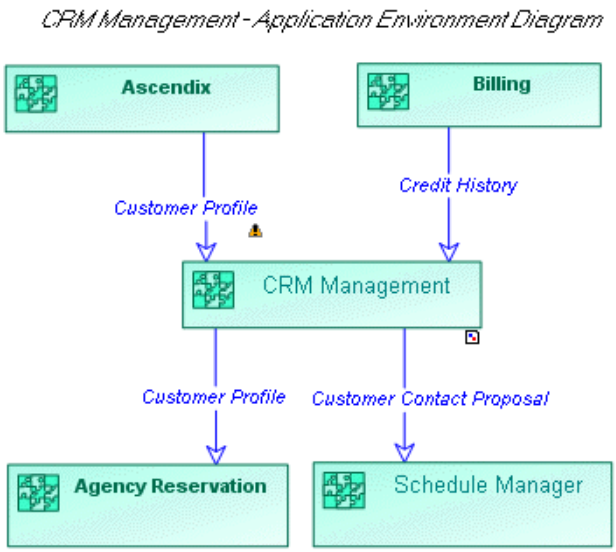
DETAILING APPLICATION INTERNAL ARCHITECTURE

HOPEX IT Architecture allows you to refine description of the internal architecture of an application. Having created the environment diagram of an application which describes its exchanges with other applications, you can describe internal operation of this application in detail.

Example of "CRM Management" application architecture environment diagram.

IS management of Voyages and Vacations installs a Customer Relations Management (CRM) application.

An "Ascendix" software package sends customer profiles to the Customer Relations Management application, which makes these profiles available to the Agency Reservation application. The history of credit granted to customers is sent to the CRM application by the Billing application, and contract proposals are sent to the Sales Schedule Manager.



In the application internal architecture description, we shall show how message contents received and sent by the application are processed by its components.

To automatically ensure consistency between these two description levels, **HOPEX** proposes representation of communication points with the exterior by means of roles.

Components of the described application (sub-applications or services) send and receive these messages to and from these roles. **HOPEX** therefore automatically represents the external applications, org-units, etc. that send and receive messages with the same content as roles.

To facilitate reuse of message contents, **HOPEX** proposes a list of candidate contents when creating a message that has a role as sender or receiver. All message contents sent or received by the described application, for example in an environment diagram, are therefore proposed in the message creation dialog box.

Example of "CRM Management" Application Internal Architecture Diagram.

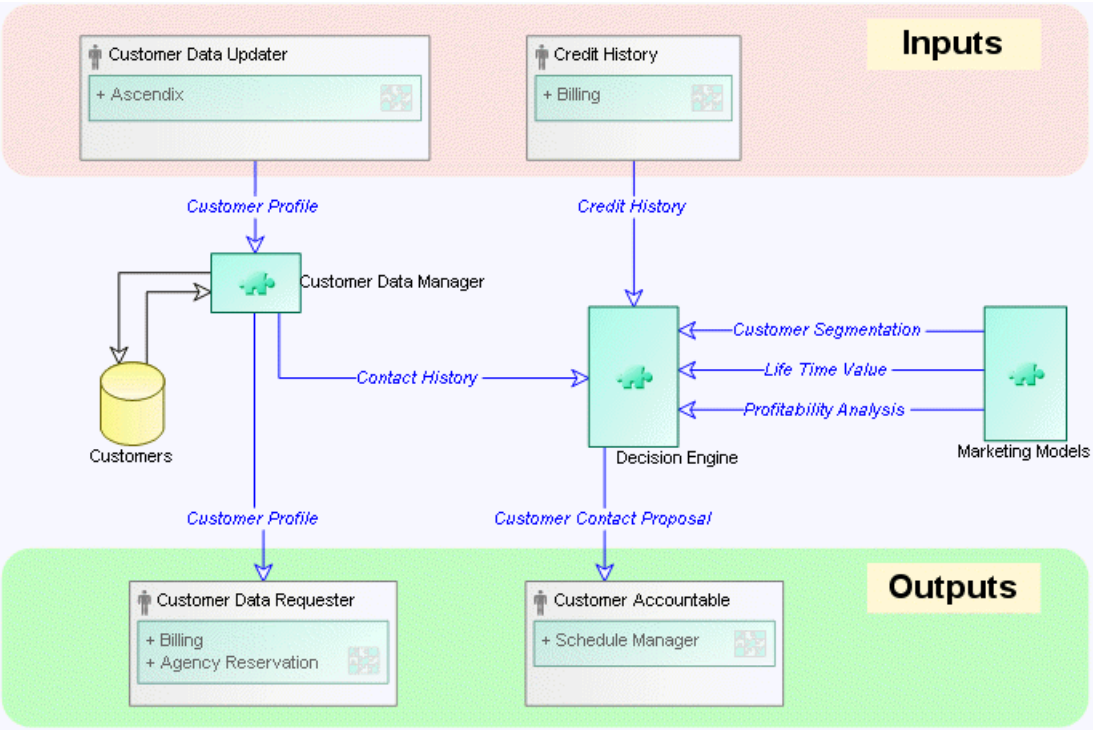
Three services are managed by the "CRM Management" application.

- The Customer Data Manager receives customer profiles and makes these available to user applications.

- The Decision Engine uses the customer credit history to prepare a contract proposal.

- The Marketing Models service provides data required by the Decision Engine.

The sender and receiver applications that exchange flows external to the "CRM Management" application automatically appear in the roles.



CRM Management - Application Internal Architecture Diagram

For more details on diagrams and described objects, see:

✓ "Detail of concepts used", page 127

Opening an Application Internal Architecture Diagram

To open the internal architecture diagram of an application:

1. In the **Main Objects** navigation window, select the **Applications** folder.
2. Expand the application folder.
3. Double-click **Application Internal Architecture Diagram**.

Describing Message Flows

You can define data flows exchanged between the application and the exterior in application architecture diagrams.

☛ *For more details on creation of flows and their content, see "Using flows", page 15.*

Running the Check Tool (MEGA Windows Front-End)

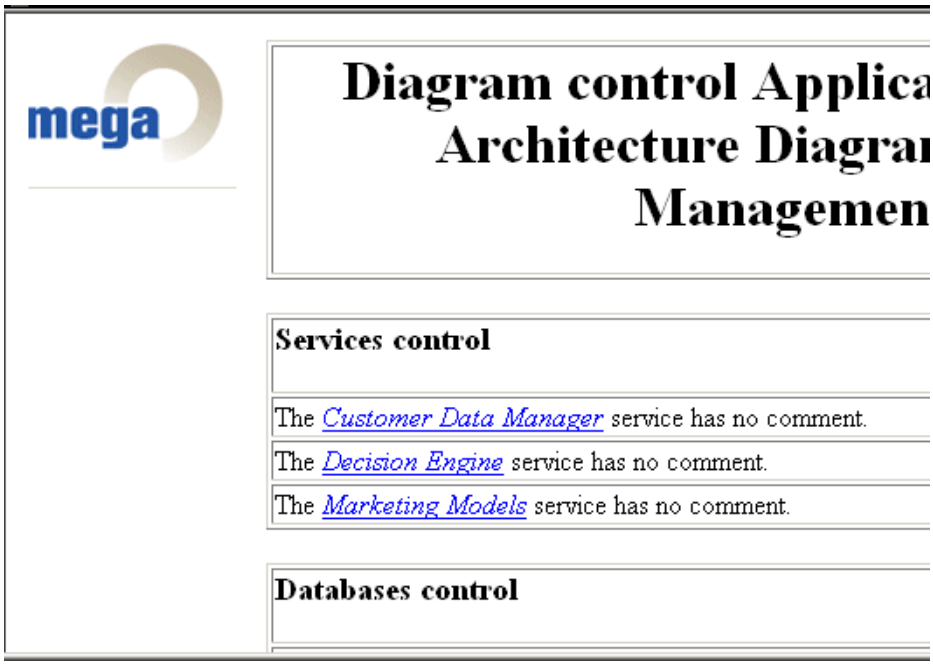
Checking diagram objects

HOPEX allows you to check the objects in your diagram..

To run the check tool:

1. From the menu bar of your **HOPEX IT Architecture** desktop, select **Diagram** > **Check** > **Check Descriptor**.
   A dialog box asks you to select a check type.

2. Select "Check of an application architecture diagram" and click **OK**.
An HTML page opens, detailing checks carried out.



Objects with no comment are indicated. When you click on the link representing the name of the checked object, the explorer opens and allows you to carry out the necessary modifications.

When the diagram has been completed, you can again produce its report (MS Word).

> ☛ *For more details on production of reports (MS Word), see the **HOPEX Common Features** guide, chapter "Generating Documentation with HOPEX", paragraph "Generating Reports (MS Word)".*

In **HOPEX IT Architecture**, the check tool is available on:

- application (AAD) or technical (TAD) architecture diagrams
- Application and Service type objects (via the pop-up menu).

Applying consistency rules

You can check your diagram by applying a regulation. A regulation is a set of rules that defines how to model objects.

To apply a regulation to a diagram:

1. Open the diagram in question.

2. From the menu bar of your **HOPEX IT Architecture** desktop, select **Diagram** > **Check** > **Regulation with Propagation**.
   The list of regulations that partition rules appears. When a regulation has been defined in the options, it is proposed by default in the list.
3. Select the regulation that you want to apply to the diagram.
4. Click **OK**.
   A report in the form of an HTML page opens. It contains results of the consistency check on diagram objects. Icons appear alongside each object indicating if a rule has been respected or not.

> ☛ *For more details on consistency rules, see the **HOPEX Common Features** guide, chapter "Generating Documentation with HOPEX", paragraph "Checking Objects".*

DESCRIBING INTERNET APPLICATIONS

Applications of Internet/Intranet type are expanding fast, mainly due to their widespread free availability. Database consultation and update applications require, in addition to development, subtle architectural modeling.

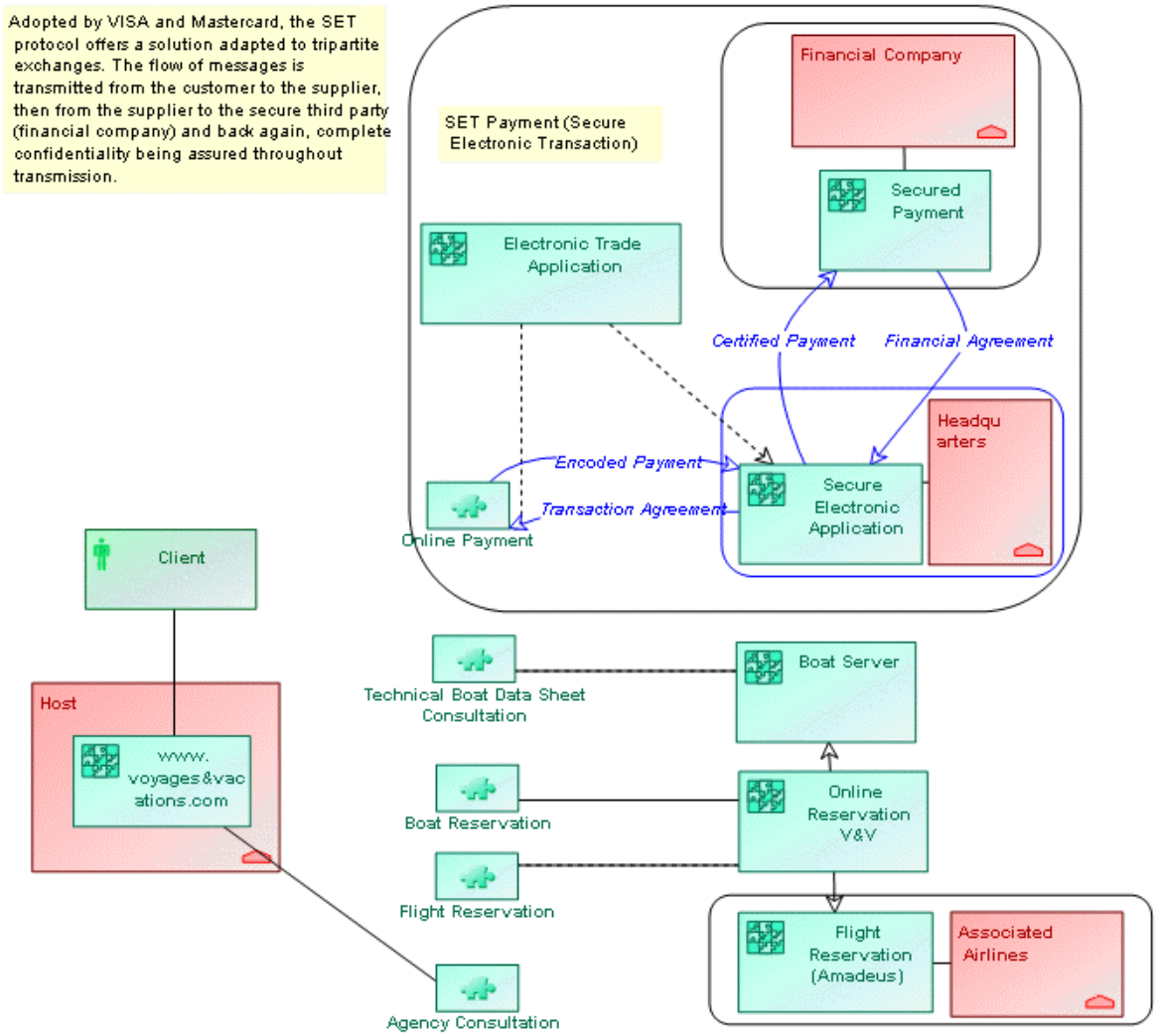
The important point is not to describe in fine detail the physical structure of applications (by nature evolutionary), but the architecture of the main components of these information systems:

1. Identify applications (conventional or Internet), their possible breakdown into sub-applications, their interactions and databases.
2. Identify sites such as hosts, duplicate sites and the org-units brought into play.
3. Identify the messages essential for conceptual understanding of the applications and the main tools used (for payment, reservation, etc.).

The conceptual and synthetic view of the applications analysis can be supplemented by a physical description, notably by using technical infrastructure diagrams.

> Example: Commercial site specification
>
> Voyages & Vacations provides a Web site for its customers that contains applications and data. It is therefore possible from the Web site to consult the catalog, make reservations and make secure payments on the Web site.
>
> The Information Systems Manager has created a diagram that describes the operation. It takes the form of an SET Internal Application Architecture Diagram.
>
> SET (Secure Electronic Transaction) is a protocol enabling secure interchanges between the customer, the enterprise

and the financial company. It has been adopted by Visa and Mastercard.



A tool is an element of an application made available to the end user.

In our example, consultation, reservation and payment tools are made available to the customer from the Web site.

Using external references you can access the Voyages & Vacations catalog and view technical specifications corresponding to each type of boat.

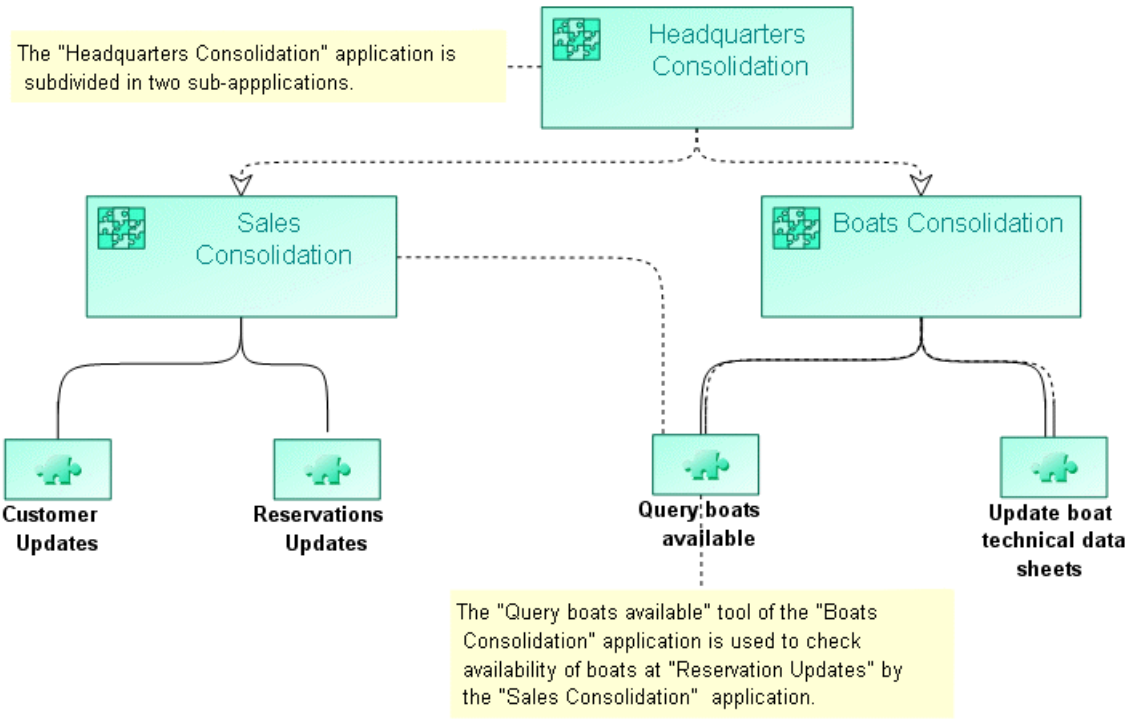
☛ *For more detailed information on creation of external references, see the **HOPEX Common Features** guide.*

For more details on diagrams and described objects, see: "Detail of concepts used", page 127.

CREATING APPLICATION TREES

Application trees are diagrams used in the same way as Application Architecture Diagrams. As a general rule, they are constructed after the event, and serve somewhat as tables of contents. In an application tree, applications and their sub-applications can be identified.

In addition, application trees enable the display (and repair) of gaps and inconsistencies in the breakdown of applications.



Creating the Application Tree of an Application

To create the application tree of an application:

1. In the **Main Objects** navigation window, right-click the application in question and select:
   - (***MEGA Windows Front-End***) **New** > **Diagram** then in the dialog box that appears, select **Application Tree**.

   Keep the "Diagram initialization" option (this option makes objects available, automatically placed in the diagram created) and click **Create**.
   - (***MEGA Web Front-End***) **New** > **Application Tree**.

2. You can create:
   - applications, see "Creating an application", page 37.
   - services, see "Creating services", page 37.
   - composition links, see "Creating composition links", page 37.
   - composition links, see "Creating architecture links", page 37.

Creating an application

To create an application:

1. In the diagram insert toolbar, click **Application** . 
2. Click on the diagram.
   The application creation dialog box appears.
3. Enter the name of the application you wish to create.
4. Click **Create**.
   The application appears in your diagram.

Creating services

A service is created in the same way as an application, using the **IT Service** button in the insert toolbar. 

Creating composition links

Composition links define the applications participating in development and deployment of an application.

To create a composition link between two applications:

1. In the diagram insert toolbar, click **Link** . 
2. In the diagram, connect the first application to the second.
   The list of elements deployed with an application is displayed in the properties dialog box of the application in question, in the **Sub-applications** section of the **Characteristics** tab.

Creating architecture links

Architecture links apply to sub-elements that are involved in operation of the application but not necessarily deployed with it.

Architecture links are not visible by default. To display these:

1. In the diagram toolbar, select **Views and Details** . 
2. In the dialog box that appears, select the **Execution** view.
3. Click **OK**.

To create an architecture link between two applications:

1. In the diagram insert toolbar, click **Link** . 
2. In the diagram, connect the first application to the second.
   The link creation dialog box appears.
3. Select "Application within Internal Architecture" link type and click **OK**.
   The architecture links appear as dotted lines.

The list of elements involved in operation of an application are available in the properties dialog box of the application in question, in the **Architecture** tab, **Application Architecture** subtab.

DESCRIBING INTER-SITE TELECOMMUNICATIONS INFRASTRUCTURE

To provide an overview of the telecommunications between the different sites of a company, you need to produce diagrams with different levels of detail, showing the physical or virtual connections used to exchange information.

Descriptions can be made by successive refinements to the desired level of detail.

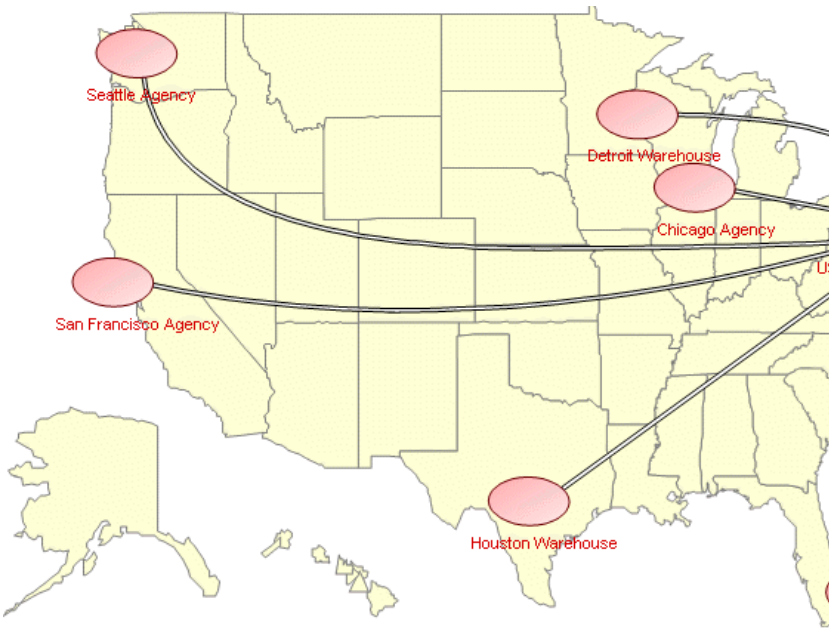
Example

Telecommunications Infrastructure

The IS manager at Voyages & Vacations has designed several diagrams providing an overview of the telecommunications infrastructure linking the different sites of the company.

The first diagram shows the methods of communication between Headquarters and the representation offices of Voyages & Vacations located outside the U.S.

The second diagram focuses on communications within the U.S., between the headquarters in New York and the travel agencies.



Consulting Existing Diagrams

To access the first map, you have to find its diagram from work already carried out by Voyages and Vacation IS Management and memorized in the repository.

To consult an existing diagram:

1. In the **Main Objects** navigation window, expand the **Technical Infrastructures** folder.
2. In the list, right-click the infrastructure and select **Technical Infrastructure Diagram**.

   ☛ *Alternative **MEGA Windows Front-End:** double-click infrastructure ().*

   The diagram opens.

Inserting Graphic Elements in a Technical Infrastructure Diagram (MEGA Windows Front-End)

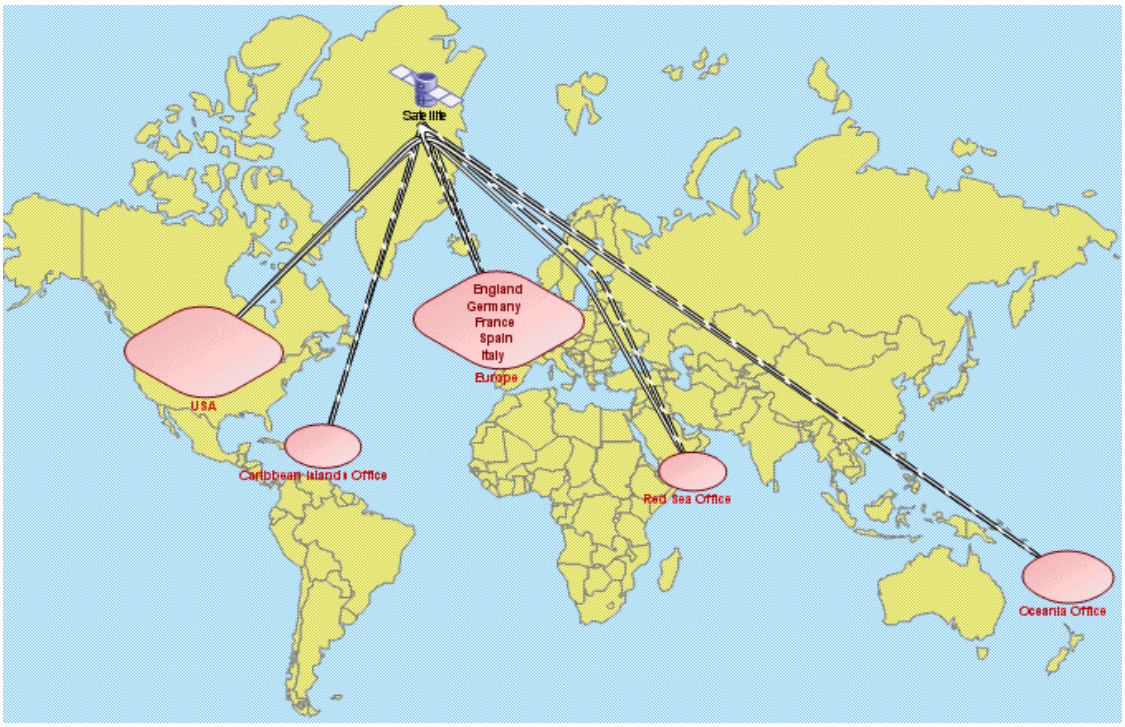
A background can be used to help understand the diagram.

```
Example: a world map.
```

To display or hide this background:

1. In the **HOPEX** menu bar **(MEGA Windows Front-End)**, select **View > Background**.

Example



HOPEX IT Architecture proposes a certain number of common graphic objects you can use, and you can add new objects from an image library.

☺ *You can also enhance a diagram using various graphic objects, imported by copy/paste.*

MODELING TECHNICAL INFRASTRUCTURES

Principles of handling Technical Infrastructure Diagrams are similar to those described above.

TIDs allow you to identify and represent hardware items (such as servers and workstations), and to indicate their characteristics (such as operating systems, processor speed, memory and hard disk capacity). The networks that connect these machines are documented in terms of topology, protocol and data transfer rate.

These diagrams indicate the machines hosting databases and applications.

Example: Headquarters TID

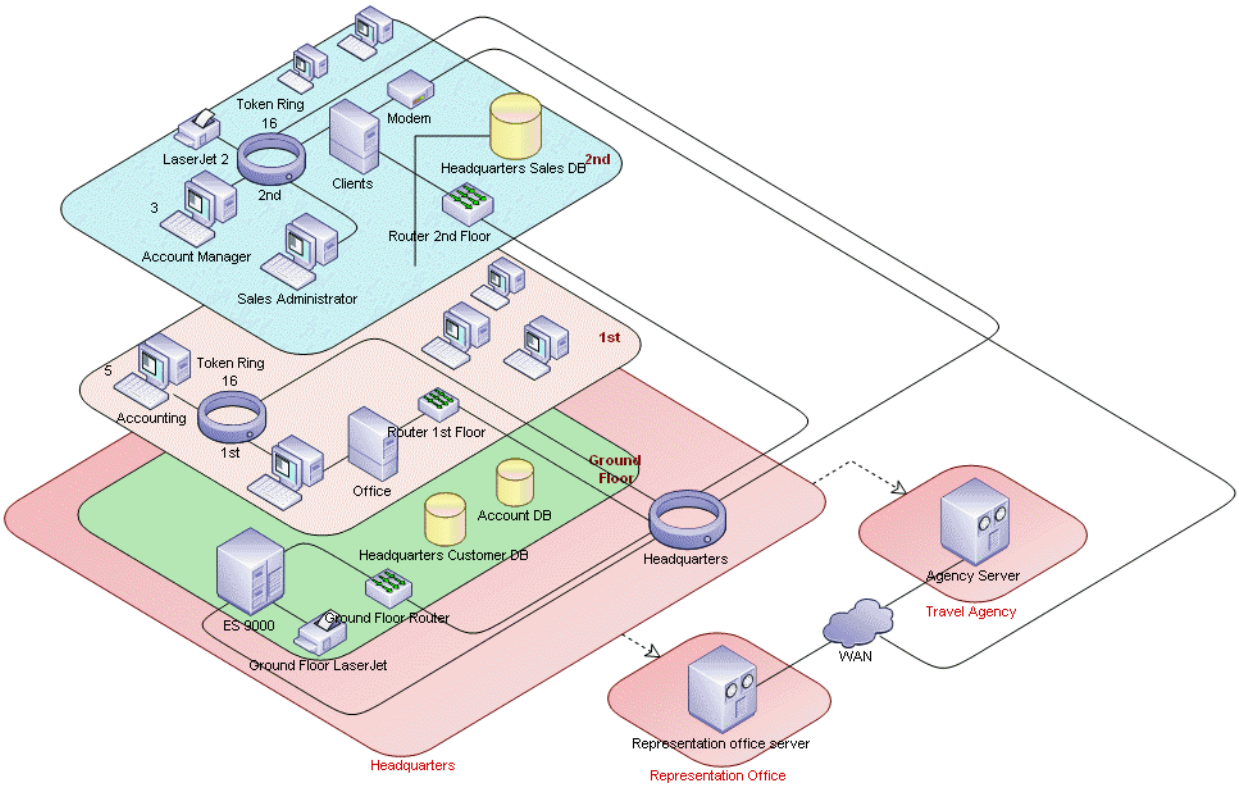
The Voyages & Vacations IS department has designed a TID for the corporate headquarters.

This diagram shows the computers found at headquarters, in particular the workstations on the 1st floor

(administrative) and the 2nd floor (where sales are consolidated).

These two LANs are interconnected.

The server on the 2nd floor manages communications with the agencies and representation offices.



Objects handled in a Technical Infrastructure Diagram are:

- sites
- servers
- applications
- databases
- workstations
- networks
- network nodes

☛ *See "Technical Infrastructure Diagram", page 133.*

You can customize your diagram. See the **HOPEX Common Features** guide, chapter "Handling Diagrams" to:

- modify the shape of an object

☛ *The shape used for certain objects, such as messages, varies according to the type assigned to them. Example: when creating a node, you are asked to select one of the following types: "Satellite", "Printer", "Hub", "Modem", "Router" and "Bridge". Similarly, after*

creating a network, you can specify its type in the characteristics tab of its properties dialog box.

- display object characteristics

  You can for example choose to display the number of workstations available for a type of org-unit (example: "Accountant").

  ☛ *These values can be modified in the **Characteristics** tab of the Properties dialog box of the object.*

  It can be of interest to configure the characteristics of other objects, for example networks, for which you can display topology, protocol and transmission flow.

- modify text field positions (**MEGA Windows Front-End**).

APPLICATION ARCHITECTURE REPORTS

offers reports that enable analysis of repository data to obtain an improved view of information.

There are several report templates, specific to analysis domain and subject. Presented here are report templates used in the framework of **HOPEX IT Architecture**.

HOPEX IT Architecture Report Templates

A report template defines the parameters and the data on which a report is based.

Prerequisite:

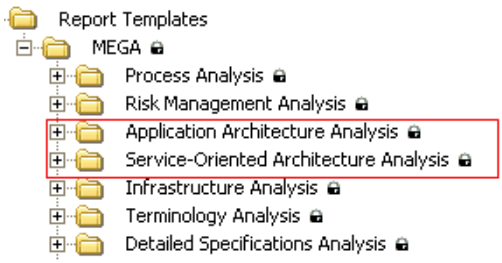
To display report templates you must have "Advanced" metamodel access.

To configure your metamodel access:

1. Access the **Options** management window:
   - ***(MEGA Windows Front-End)*** from the menu bar of your **HOPEX IT Architecture** desktop, select **Tools** > **Options**
   - ***(MEGA Web Front-End)*** from the **Miscellaneous** tool group of your **HOPEX IT Architecture** desktop, select **My Account > Options**.
2. In the options tree select **Repository** and check that **Metamodel Access** option is "Advanced".
3. Click **OK**.

To access report templates:

1. Access the **Utilities** navigation window.
2. Expand the **Report templates** folder.
   Report templates specific to **HOPEX IT Architecture** concern different domains; application architectures, service-oriented architectures (SOA), time constraints acting on the IS, etc.



Report templates linked to application architectures:

- Application Functional Analysis: an application functional analysis compares a set of elements providing functions with a set of expected functions (functional scope).
- Application Architecture Management: this report template is dedicated to study of an application architecture management. It provides details of projects and persons involved in this management.
- Infrastructure Compliance: compliance of infrastructures enables analysis of technical infrastructures involved in an information system. For example, this report template can show the usage level of a set of technologies, or compliance of a set of information sub-systems with a technical infrastructure group.
- Deployment: this report template is dedicated to the deployment problem. It provides answers to questions such as "where are applications deployed?" and "what are the differences between an application architecture and one of its deployments?".
- Project Management: this report template describes impact of a set of projects on the information system.
- As Is - To Be Architecture Analysis: this report template provides an As Is - To Be view of application architectures as a function of selected time periods.

Report templates relating to service-oriented architectures:

- Automated Process Supervision: this report template contains indicators and dashboards dedicated to supervision of automated business processes.
- Application Functional Analysis: this report template compares a set of elements providing functions with a set of expected functions (functional scope).
- Business Process Automation: this report template enables detection of potential improvements in information service support of human activities.
- IT Services Attached To Process: this report finds services relating to a set of processes.
  See "Service Oriented Architecture (SOA)", page 117.
- Impacts of IT Services on Business Processes: modifications to IT services have an impact on the business processes that use them in the context of their implementation. This tool finds all business processes impacted by a set of IT services.
- IT Services Supervision: this report template contains indicators and dashboards dedicated to supervision of IT services and applications.

Context Analysis Reports

- Time Constraint Analysis: this report template enables representation and modification of time constraints between selected objects.
- Varied Object Analysis: this report template details variations from given varied objects.
- Variant Object Analysis
- Variation Analysis: this report template enables detailing of varied objects, variants and linked variations from given variations.
  For more information on variations, see the **HOPEX Common Features** guide.

These report templates allow you to create reports with predefined parameters.

☛ *For more details on reports, see the **HOPEX Common Features** guide, "Generating Documentation".*

☛ *To create your own report templates, see the **HOPEX Power Studio** guide.*

DESCRIBING INTERACTIONS

The following points explain how to describe the interactions between the applications of an enterprise and those of its partners.

USE CONTEXT

You can use interactions to provide a detailed description of how to automate exchanges between an enterprise and external org-units.

These exchanges may be described using standards defined by international groups such as the OAG (Open Applications Group) or by marketplace providers.

An enterprise can also define its own interaction modes, with subsidiaries or subcontractors for example.

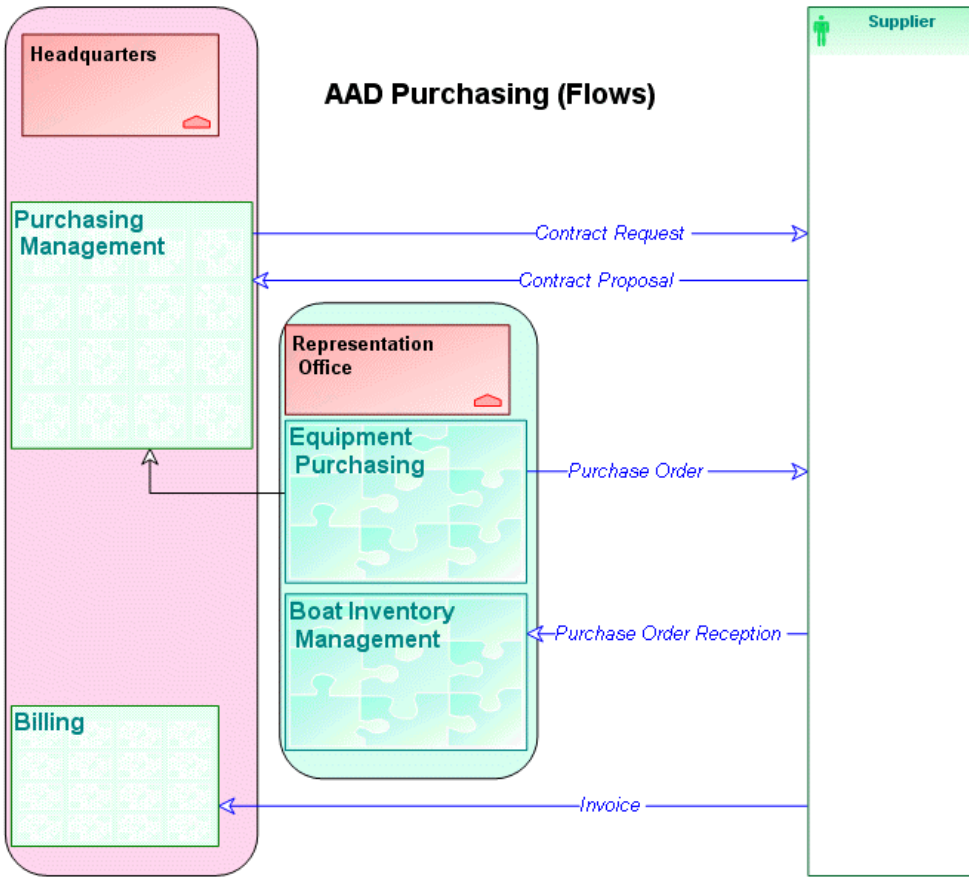
The following example illustrates how to describe exchanges between an enterprise and its suppliers. It uses the "Purchasing Management System" application.

Purchasing Management System Internal Architecture Diagram

The "Purchasing Management System" is described by two **Internal Application Architecture Diagrams**.

The "Purchasing Management System - AAD Purchasing (Flows)" diagram presents:

- the main applications that intervene in the purchasing management system.
- the main exchanges with suppliers in the context of the purchasing management system.



Equipment purchases for boat maintenance are made directly by the Representation Office, using the "Equipment Purchasing" application. This application calls a "Purchasing Management" application located centrally at Headquarters.

If the purchase involves a new type of equipment or a new supplier, a contract request is sent to the supplier using

the application at Headquarters. The supplier returns a contract proposal.

In the other cases, a representative from the Representation Office sends a purchase order to the supplier using the "Equipment Purchasing" application.

Reception of purchases delivered from the supplier is handled by the "Boat Inventory Management" application at the Representation Office.

The invoice sent by the supplier is managed in the "Billing" application at Headquarters.

☛ *You can display activitites corresponding to applications, see* *"Displaying Activities corresponding to Applications", page 52*

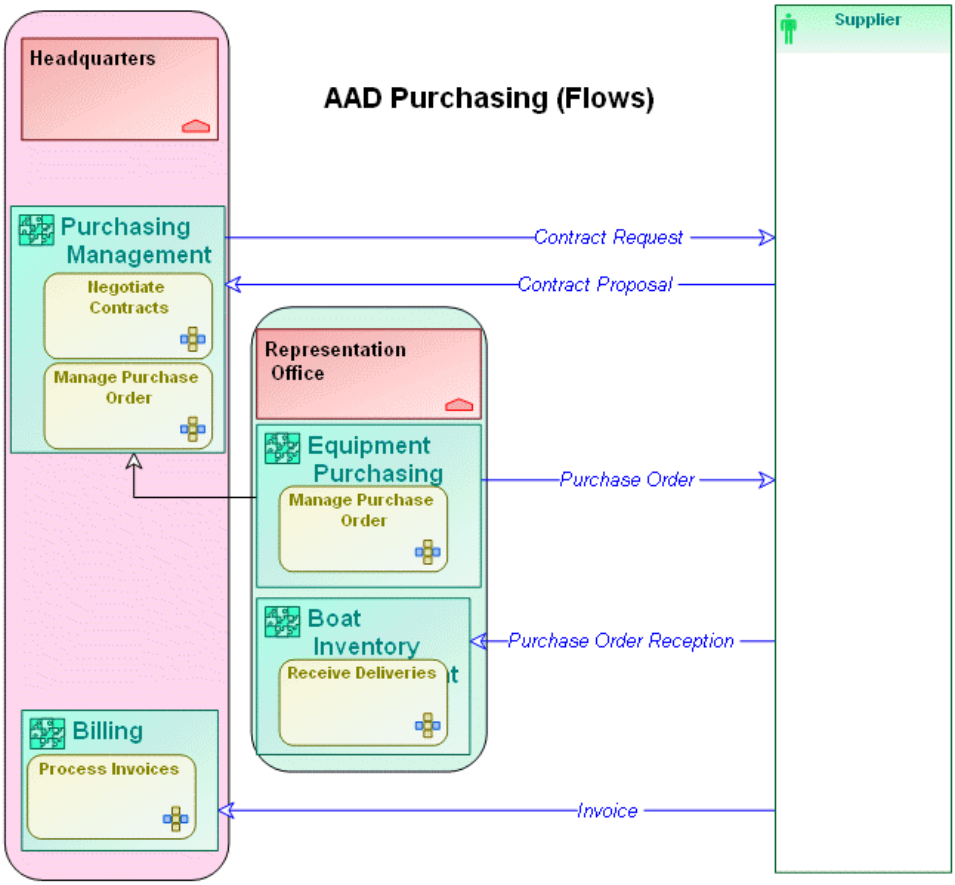
Displaying Activities corresponding to Applications

☛ *If you also have **HOPEX Business Process Analysis**, you can display activities corresponding to applications.*

To display activities corresponding to Applications

1. In the diagram toolbar, select **Views and Details** .

2. Select the **Business Functions, Activities** view.



The "Manage Purchase Order" activity is divided between the "Purchasing Management" application at Headquarters and the "Equipment Purchasing" application at the Representation Office.

The "Negotiate Contracts", "Receive Deliveries", and "Process Invoices" activities are handled by the "Purchasing Management", "Boat Inventory Management", and "Billing" applications respectively.

☛ *Implementing interactions by activities of a process is described in section "Automating Exchanges: Interactions", page 53.*

Automating Exchanges: Interactions

📖 *An interaction represents a contract established in a specific context between autonomous entities that are internal or external to an enterprise. These entities can be enterprise org-units, applications,*

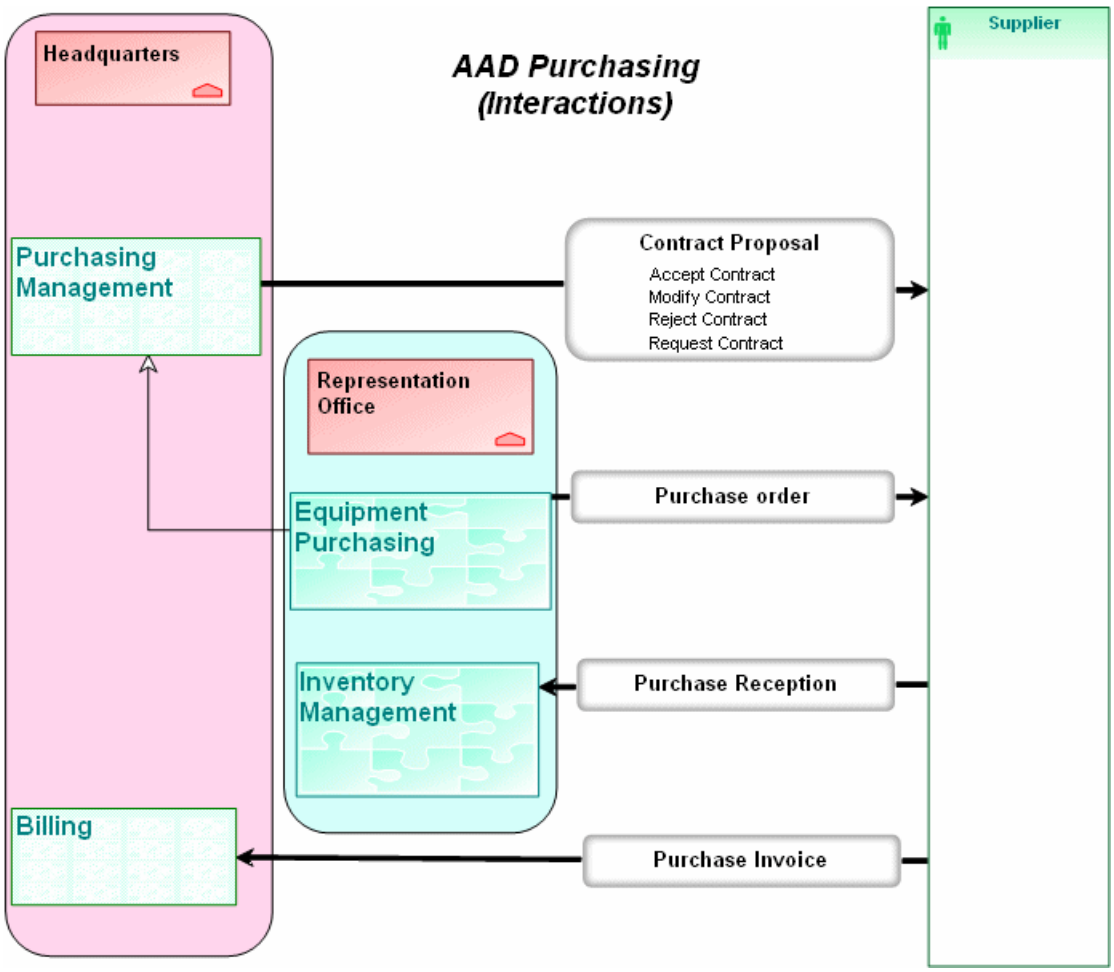
activities or processes, as well as external org-units. The content of this contract is described by an exchange contract.

An exchange contract is a model of a contract between organizational entities. This contract is described by exchanges between an initiator role and one or several contributor roles.

Representing the flows exchanged between enterprise applications and suppliers may be sufficient for a general understanding of the relations between the enterprise and its suppliers.

It is not sufficiently detailed, however, for automating the exchanges with suppliers.

The **Application Internal Architecture Diagram** below presents exchanges in the form of interactions. The interaction allow us to describe in detail the actual interaction between the two partners.



DESCRIBING INTERACTIONS

Creating an interaction with an Existing Exchange Contract

To create an interaction:

1. In the diagram objects toolbar, click **Interaction**  .
2. Draw a link between the two entities in communication.
   The **Add Interaction** dialog box appears.
3. Click the arrow at the right of the **Exchange Contract** field and select **Create Exchange Contract**.
4. Enter the **Name** of your exchange.
5. Click **OK** to close this dialog box.
   The exchange is automatically created.
6. Click **OK**.
   The interaction appears in the diagram.

Describing Exchange Contracts

An **Exchange Contract** can be supported by **Exchanges** or **Exchange Contracts** representing information exchanges between architecture components.

> *An exchange contract is a model of a contract between organizational entities. This contract is described by exchanges between an initiator role and one or several contributor roles.*
>
> *An exchange use represents the usage of an exchange in another exchange contract.*
>
> *An exchange contract use represents the usage of an exchange contract in another exchange contract.*

To describe that an exchange is used by an exchange contract:

1. Open the exchange contract properties dialog box.
2. Click the **Exchange** tab.
3. Click the **New** button.
   A selection dialog box opens.
4. Select **Exchange Use**, which is the type of exchange you want to use, and click **OK**.
   The creation dialog box opens.
5. Click the arrow at the right of the **Specification** box.
6. Select **List** in the drop-down list and select the exchange to be associated with the exchange use.
   The name of the exchange appears in the **Specification** field.
7. In the **From** field, select the described exchange role connected to the Consumer role of the exchange used.
8. In the **To** field, select the described exchange role connected to the supplier role of the exchange used.

9. Click **OK**.

   ☛ *You can associate several exchanges with the exchange contract.*

10. Click **OK**.

Describing Exchange Message Flows

Content of the exchange is described by message flows and their content which are exchanged between the two roles representing the stakeholders in the conversation.

> 🕮 *A message flow is information flowing within an enterprise or exchanged between the enterprise and its business environment. A message flow can carry a content.*
>
> 🕮 *The content designates the content of a message or an event, independent of its structure. This structure is represented by an XML schema linked to the content. A content may be used by several messages, since it is not associated with a sender and a destination. There can be only one content per message or event, but the same content can be used by several messages or events.*

To describe message flows exchanged:

1. Open the properties of the Exchange.
2. Select the **Message Flows** tab.
3. Click the **New** button.
   The **Creation of Message Flows With Content** dialog box opens.
4. From the **Content** drop-down list, select the content you wish to associate with the message flow.
   The message flow with its content is displayed in the list of conversation contents.

   ☛ *You can associate several contents with the message flow.*

5. Specify the direction of each message flow.
6. Click **OK**.

MODELING COMPLEX INFRASTRUCTURES

HOPEX IT Architecture allows you to further detail system architecture description to take into account systems using resources other than software. For example, a system can comprise radar, pylons, various software elements, a network, people, etc. To define heterogeneous systems, **HOPEX IT Architecture** provides three diagrams, the Resource Architecture Tree, the Resource Architecture Diagram and the Artifact Assembly Diagram. They enable modeling of system components as well as their communication mode.

The following points are covered here:

☛ *Modeling functionalities of complex infrastructures are available when **Infrastructure Modeling Post 2009 SP2** is selected (in **HOPEX Options**, **Business Process and Architecture Modeling** folder).*

CONTEXT OF USE

Functionalities proposed by **HOPEX IT Architecture** for modeling complex infrastructures enable representation of equipment and organizational resources required for system operation, interactions between components, communication means supporting these interactions, and services offered and used by the modeled architecture.

The components described can follow modeling methodology that is ascending (from most detailed to most conceptual) or descending (from most conceptual to most detailed).

Presentation of these functionalities is based on the example of a support center, presented using a descending approach from resource architecture diagrams and an artifact assembly diagram.

Resource architecture environment diagram

Resource architecture environment diagram

A resource architecture environment presents an resource architecture use context. It describes the interactions, between the resource architecture and its external partners, which allows it to fulfill its mission and ensure the expected functionalities.

```
The support center provides a service to the clients. An
external partner provides technical expertise for some
questions.
```

The diagram includes:

- An *architecture use* element that represents the Resource Architecture internal to the environment.

  An architecture use is the installation of a resource architecture within another resource architecture or an environment.

- A *partner resource architecture* that represents a resource architecture used in the context of the described resource architecture environment.
- A *resource architecture user* component that represent the user category of services provided by the environment.

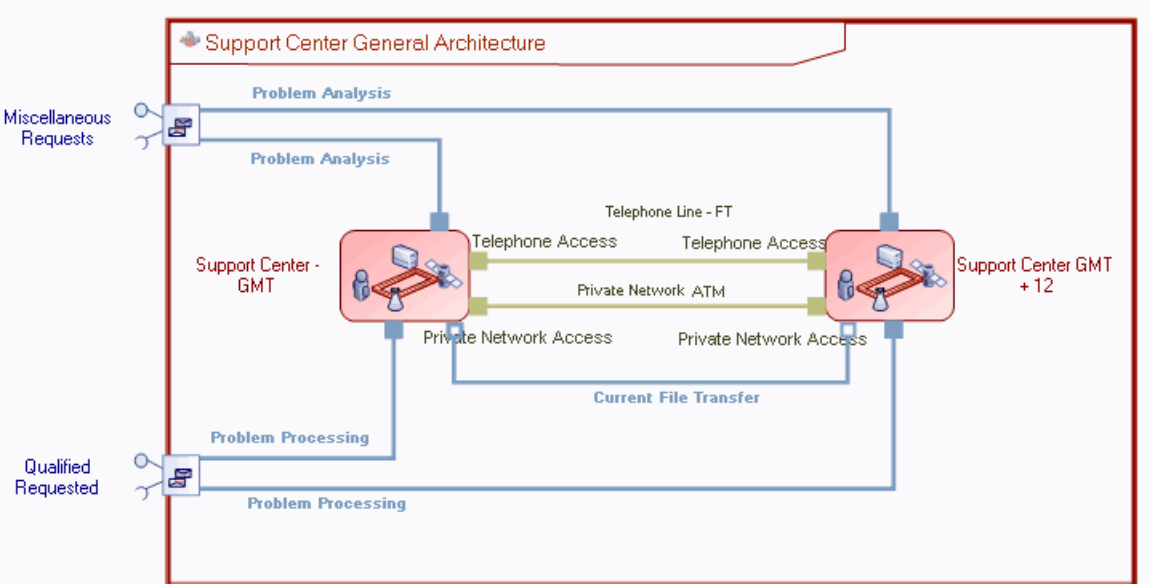
  A resource architecture user represents an organizational unit interacting with the boundaries of a resource architecture environment.

- *Interactions* between the components representing requests for services.

  An interaction represents a contract established in a specific context between autonomous entities that are internal or external to an enterprise. These entities can be enterprise org-units, applications, activities or processes, as well as external org-units. The content of this contract is described by an exchange contract.

Resource architecture diagram

The following diagram describes the support center environment.



24/24 cover is assured by two "Support Center" teams, one in Paris (GMT Support Center) and the other in Singapore (GMT+12 Support Center). Each of these assures 12 hour support service.

At the end of its service period, each center transfers dossiers to the center taking over.

One support service handles customer requests in real time, while a second support service is assured offline for requests already qualified.

The components of the *resource architecture* are described in a resource architecture diagram.

> 🕮 *A resource architecture is the combination of physical and organizational assets configured to supply a capability.*

The diagram includes:

- two elements of *Architecture Use* type representing the two support centers.

  > *An architecture use is the installation of a resource architecture within another resource architecture or an environment.*

- two *Service Points* representing the two support services offered by each of the centers.

  > *A service point is a point from which the system receives a service request from another system, and provides the service requested.*

- *Interactions* between the service points and the centers, representing service requests on the one hand and transfer of current dossiers on the other.

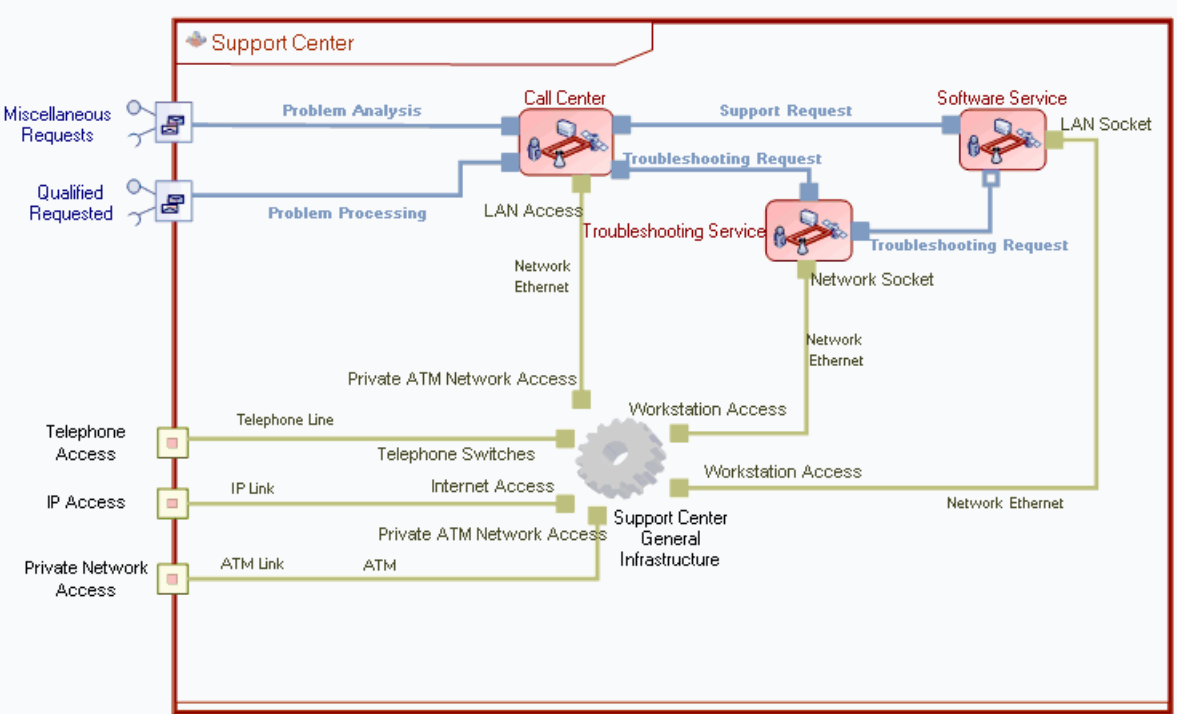
  > *An interaction represents a contract established in a specific context between autonomous entities that are internal or external to an enterprise. These entities can be enterprise org-units, applications, activities or processes, as well as external org-units. The content of this contract is described by an exchange contract.*

- *Communication Channels* between the two centers.

  > *A communication enables physical connection between two equipment resources. It supports interactions that define communication between these resources. Communication channels connect resources with the exterior via communication ports.*

Support Center Architecture

From the general architecture of the system, you can access the resource architecture diagram that describes the structure common to the two support centers.



The call center handles all requests, qualified or not, checks these and sends them either to software support or to the troubleshooting service.

If software support detects an operating problem due to equipment, it records the incident and alerts the troubleshooting team.

The three structures depend on the same general hardware infrastructure.

This diagram includes the following equipment resources:

- *Physical Assets*.

  📖 *A physical asset represents implementation of an artifact in a resource architecture.*

- *Communication Ports* which represent physical communication points of the equipment architecture and which implement the service points.

  📖 *A communication port is a physical point of communication with a resource. These conform to specific communication protocols. A communication port implements service and requests points.*

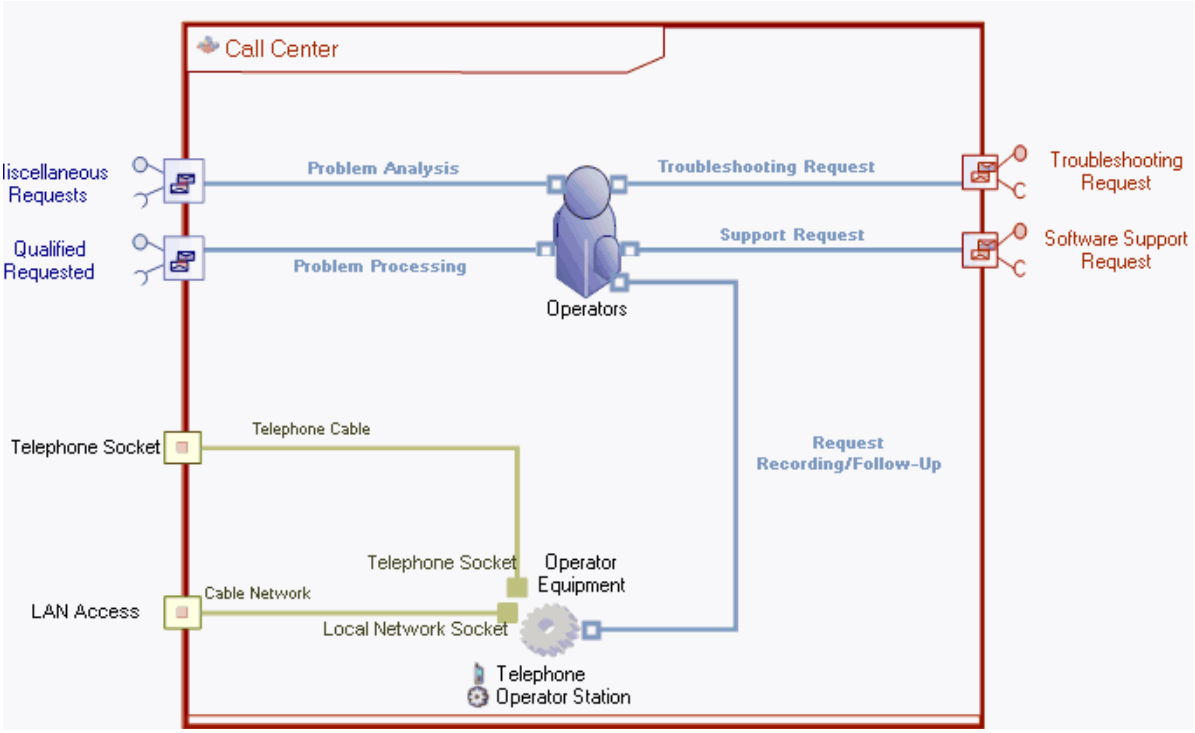
- *Communication Channels* between the two centers.

  📖 *A communication enables physical connection between two equipment resources. It supports interactions that define*

communication between these resources. Communication channels connect resources with the exterior via communication ports.

Call center architecture

The resource architecture diagram of a call center describes the equipment and organizational resources required for handling service requests.



A team of operators handles all requests, whatever their nature, by telephone or by e-mail.

The operator identifies the caller, records the request, applies a first filter (in case of error) and transmits the call to software support or troubleshooting support as appropriate via request points.

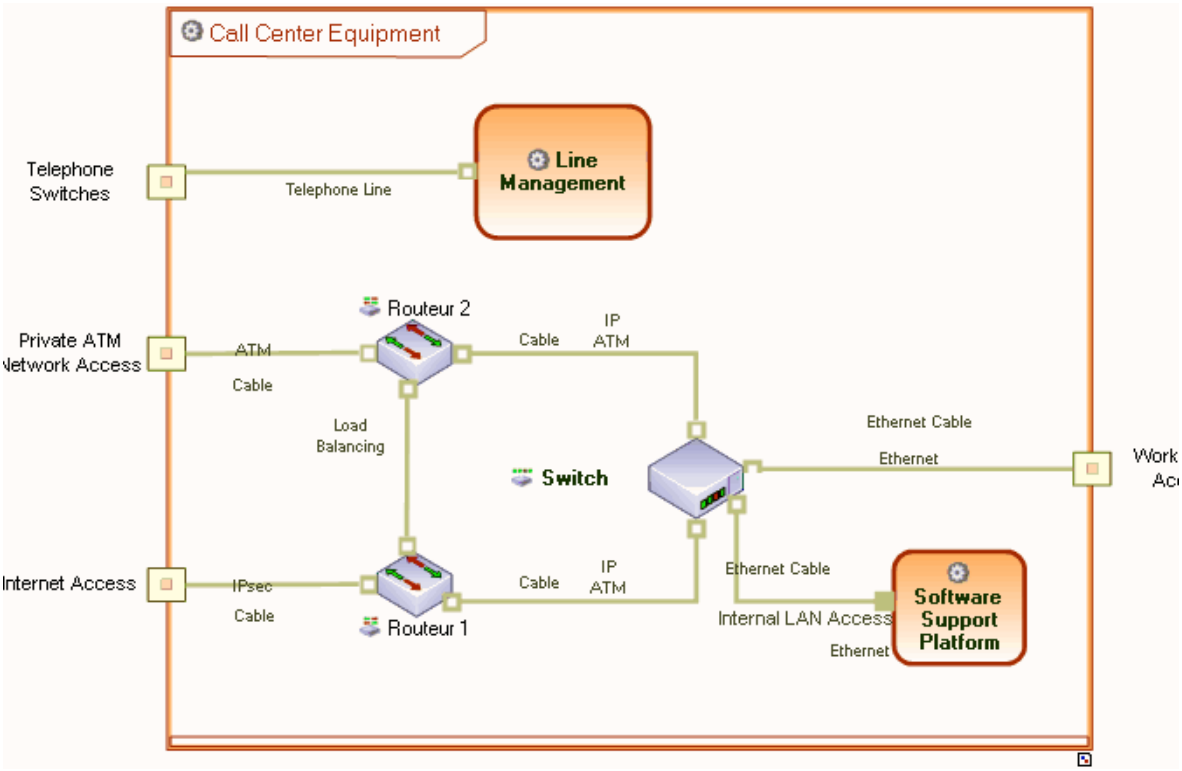
We assume for simplification that each team comprises a manager and various members, for whom we do not describe specialization level or duty time slot.

This diagram contains two *Request Points* from which the operators make service requests to other resource architectures.

A request point is a point from which the system sends a request to another system, and receives the service requested.

Artifacts Example

Artifact assembly diagram of a support center.




```
Basic equipment architecture of a call center includes
three link points with the exterior: a telephone link, a
link to a private ATM network and an internet access link.
The workstations access point enables connection of
operator, support and troubleshooting stations.
```

This diagram includes two *Artifact Components* representing a telephone switchboard and a test platform used for software support.

> 🕮 *An artifact component represents installation of an artifact within another.*

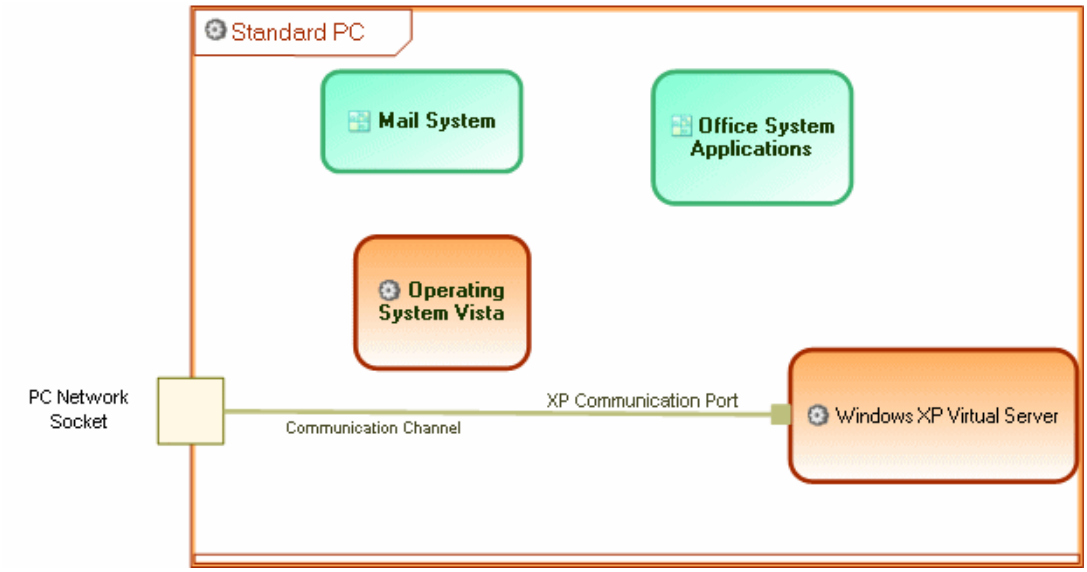
Note that the *Communication Protocols* used are specified on the communication channels. They are also specified on the communication ports.

> 🕮 *A communication protocol is a set of standardized rules for transmission of information (voice, data, images) on a communication channel. The different layers of protocols can handle detection and processing of errors, authentication of correspondents, management of routing.*

Routers are predefined artifacts, proposed in the **HOPEX IT Architecture** library.

Artifact assembly diagram of a standard PC.

The assembly diagram of a standard PC is shown below.

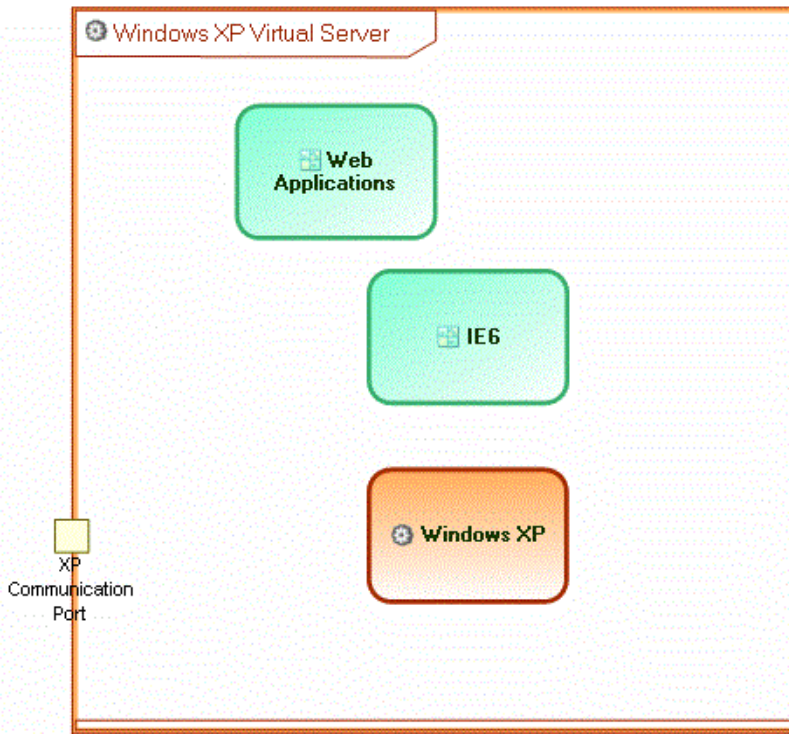


Standard PC assembly diagram

A standard PC is equipped with a Vista operating system on which office and electronic mail applications are installed.

Migration to Windows Vista causes a Web application compatibility problem with the navigator. A standard PC is

therefore connected to a Windows XP virtual server on which Internet Explorer 6 and its Web applications operate.



Windows XP virtual server assembly diagram (call centers)

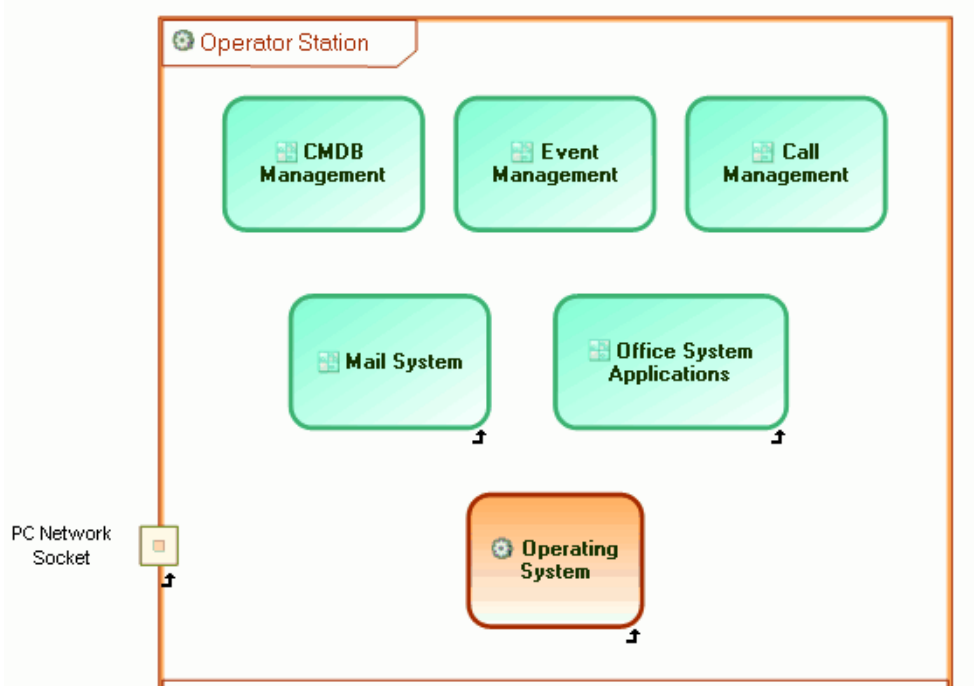
These diagrams presenting technical architecture contain components of *application host* and *database host* type representing the applications and servers installed on a standard PC.

> 🕮 *An application host represents implementation of an application in an artifact.*
>
> 🕮 *A database host represents hosting of a database in an artifact.*

Operator stations and support stations are variants of the standard station. All operator and support stations therefore have the same basic architecture, comprising:

- an operating system, Windows Vista
- an electronic mail service, Microsoft Outlook
- an office system software suite, Microsoft Office
- a PC network socket.

An operator station is based on this architecture, and in addition hosts specific applications for management of CMDB, calls and incidents.



Operator station assembly diagram

Since "Operator Station" and "Support Station" artifacts are defined from the "Standard PC" artifact, if the electronic mail service is replaced by another in "Standard PC", it is automatically modified in the operator and support stations.

☛ *For more details on variations, see the **HOPEX Common Features** guide, "Handling Repository Objects", "Object Variations".*

DESCRIBING A RESOURCE ARCHITECTURE ENVIRONMENT

> *A resource architecture environment presents an resource architecture use context. It describes the interactions, between the resource architecture and its external partners, which allows it to fulfill its mission and ensure the expected functionalities.*

A *resource architecture* comprises equipment and organizational resources required for operation of a complex infrastructure. Communications between these components are represented by interactions and the equipment means supporting these interactions are the communication channels.

> *A resource architecture is the combination of physical and organizational assets configured to supply a capability.*

Services offered by the system to its users are represented by service points. Service points are physically supported by communication ports that enable access to communication means of the system.

Creating a resource architecture environment

To create a resource architecture environment from a library:

1. Check in the **HOPEX** options, **Business Process and Architecture Modeling** folder, that the **Infrastructure Modeling Post 2009 SP2** option is selected.
2. Right-click the library that interests you and select **New > Model building block**.
   The **Choose MetaClasse** properties dialog box opens.
3. Select **Resource architecture environment**.
   The **Creation of a Resource architecture environment** window opens.
4. Enter the environment **Name** and click **OK**.
   The new resource architecture environment appears in the navigation tree.

The procedure to create a resource architecture environment diagram varies slightly depending on whether you are in MEGA Windows Front-End or MEGA Web Front-End.

HOPEX Web Front-End

To create a resource architecture environment diagram:

- Right-click the resource architecture environment and select **New > Diagram > Resource Architecture environment Diagram**.
  The diagram opens.

HOPEX Windows Front-End

To create a resource architecture environment diagram:

1. Right-click the resource architecture environment and select **New** > **Diagram**.
   A dialog box opens.
2. Select the "Resource Architecture environment Diagram" diagram type.
3. Leave the **Diagram Initialization** option selected.
4. Click **Create**.

Creating a Resource Architecture

To create a resource architecture from a resource architecture environment:

- Right-click the resource architecture environment that interests you and select **New > Resource Architecture**.

The procedure to create a diagram varies slightly depending on whether you are in MEGA Windows Front-End or MEGA Web Front-End.

HOPEX Web Front-End

To create a resource architecture diagram:

- Right-click the resource architecture and select **New > Diagram > Resource Architecture Diagram**.
  The diagram opens.

HOPEX Windows Front-End

To create a resource architecture diagram:

1. Right-click the resource architecture and select **New** > **Diagram**.
   A dialog box opens.
2. Select the "Resource Architecture Diagram" diagram type.
3. Leave the **Diagram Initialization** option selected.
4. Click **Create**.

Implementing Capabilities

Resource architecture creation can answer a need for a capability.

A capability defines an expected skill, for example "Provide Level 3 Support".

You can connect a resource architecture to a capability to describe all physical and organizational means set up to provide this service.

To indicate that a resource architecture is connected to a capability:

1. Open the properties dialog box of the resource architecture.
2. Click the **Characteristics** tab.
3. In the **Configured Capability** frame, click the **Connect** button.
   ☛ *You can also create a capability from this dialog box.*
4. Select the capability in question and click **OK**.
   The name of the capability appears in the frame.

Describing Resource Architecture Organizational Resources

Creating an architecture use

To describe that an architecture, such as a support center, uses another architecture such as a troubleshooting service, you will create in the "Support Center" a component of *Architecture Use* type, and associate with it the other architecture used ("Troubleshooting Service").

> 🕮 *An architecture use is the installation of a resource architecture within another resource architecture or an environment.*

To create an **Architecture Use**:

1. In the resource architecture diagram objects toolbar, click **Architecture Use** . 
2. Click in the described architecture frame.
   A creation dialog box asks you to select a **Deployed Resource**.
   This is the resource architecture used (for example "Troubleshooting Service"). You can select an existing resource architecture or create a new one.

   > ☛ *In the case where the resource architecture you want to use does not yet exist in the repository, you can postpone creation of this resource. If you click **OK**, the architecture use component is created without a resource.*
3. Click **Finish**.

Creating a human asset

To describe that an architecture such as a call center uses operators to take calls and handle requests, you will create components of *Human Asset* type and associate the "Operator" org-unit.

> 🕮 *A human asset is an organizational resource configured to support certain capabilities in the framework of a resource architecture.*

To create a human asset:

1. In the resource architecture diagram objects toolbar, click **Human Asset** . 
2. Click in the frame of the resource architecture diagram.
   A creation dialog box asks you to select a deployed resource.
3. Select the org-unit you wish to assign to the resource and click **Finish**.

   > ☛ *You can create a human asset without attaching an org-unit. If the desired org-unit does not exist in the repository, or is as yet unidentified, you can postpone assignment of an org-unit to the resource.*

Describing Equipment Resources

Creating a physical asset

To describe that an architecture such as a software support center relies on equipment resources such as workstations hosting applications, you will create components of *Physical Asset* type and associate the required workstations with these.

> 🕮 *A physical asset represents implementation of an artifact in a resource architecture.*

To create a physical asset:

1. In the diagram objects toolbar, click **Physical Asset**. 
2. Click in the diagram frame.
   A creation dialog box asks you to select a deployed resource.
3. Select the artifact you wish to assign to the physical asset (in this case operator stations) and click **OK**.

> ☛ *You can create a physical asset without connecting an artifact. If the desired artifact does not exist in the repository, or is as yet unidentified, you can postpone assignment of the resource to the physical asset.*
>
> ☺ *You can also create a physical asset by selecting an artifact in the **HOPEX** navigation window, and dragging it into the diagram using the mouse.*

Channels and communication ports

In a resource architecture, communication channels support transfer of information from one physical asset to another. For more details on creation of these channels and the associated communication protocols, see "Communication channels", page 79.

Communication ports enable connection of resource architecture physical assets with external equipment elements. For more details on how to connect communication ports between different components, see "Connecting Communication Element Points", page 80.

Describing Resource Architecture Services and Requests

A resource architecture is created to assure one or several services.

> ☛ *For more details, see "Service points", page 75 and "Request points", page 76.*

Describing Interaction in a Resource Architecture Diagram

In a resource architecture diagram, *Interactions* enable representation of exchanges between organizational entities.

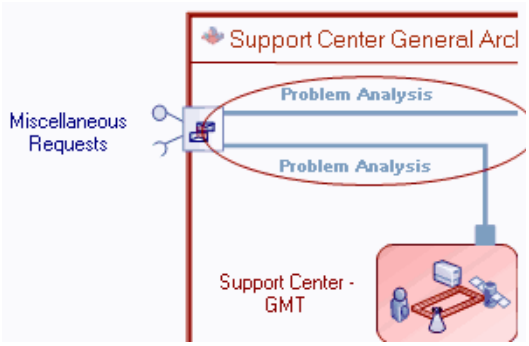
> *An interaction represents a contract established in a specific context between autonomous entities that are internal or external to an enterprise. These entities can be enterprise org-units, applications, activities or processes, as well as external org-units. The content of this contract is described by an exchange contract.*

Exchange terms are defined by an *Exchange contract* assigned to the interaction.

> *An exchange contract is a model of a contract between organizational entities. This contract is described by exchanges between an initiator role and one or several contributor roles.*

You can define interactions between:

- Two components of resource use type to represent exchanges between these entities,
- A component of resource use type and a physical asset to represent terms of use of the equipment resource by the organizational resource. For example, you can represent that software test platform use is arranged by booking.
- Two components of physical asset type to represent terms of use of a resource by another in the context of the modeled resource architecture. For example, terms of use of replacement equipment (spare parts) depends on priority level of the organization requesting this.
- A service point and one or several resource use type components to represent implementation of the service within the resource architecture,
- A component of resource use type and a request point to represent that the entity calls a resource of an external organization.



In the example of the general support system, two interactions are used to indicate that support services are handled by one center or another depending on time slot. The two interactions are executed according to the same exchange contract.

For more details on interaction management terms, see "Managing Interactions", page 77.

DESCRIBING AN ARTIFACT

An artifact is any type of physical element outside the application or organizational domain (organizational including persons). An artifact can represent a hardware system, sub-system, platform or component, or simply a physical element with specific characteristics.

You can describe components of an *artifact* in an artifact assembly diagram.

You can insert in this diagram:

- artifacts,
- application hosts,
- communication ports and channels,
- service and request points,
- interactions.

Creating an Artifact

To create an artifact from a library:

1. Check in the **HOPEX** options, **Business Process and Architecture Modeling** folder, that the **Infrastructure Modeling Post 2009 SP2** option is selected.
2. Right-click the library and select **New** > **Artifact**.

To create the diagram of an artifact ***HOPEX Windows Front-End***:

1. Right-click the artifact and select **New** > **Diagram**.
2. In the dialog box that opens, select **Artifact Assembly Diagram** and click **Create**.
   The diagram opens.

> ☛ *To create an artifact assembly diagram with **HOPEX Web Front-End**, right-click the artifact and select **New > Diagram > Artifact Assembly Diagram**.*

Describing Artifact Components

Creating an artifact component

To describe that an artifact, such as equipment architecture of an operator, implements another artifact, such as an operator station, you will create an *Artifact Component* in the equipment architecture and associate an operator station with it.

> 🕮 *An artifact component represents installation of an artifact within another.*

To create an artifact component:

1. In the artifact assembly diagram objects toolbar, click **Artifact Component** .

2. Click in the described object frame.
   A creation dialog box asks you to select the **Artifact Used**, either from existing artifacts, or from a new artifact.

   ```
   Example: select "Operator Station".
   ```

   ☛ *In the case where the artifact you wish to use does not yet exist in the repository, you can postpone its creation. If created, you can modify it later.*
3. Click **OK**.

Creating an application host

To describe that an artifact such as a standard workstation is a standard PC hosting basic applications such as an electronic mail service application, you will create an **Application Host** type component and associate the required application with it.

To create an Application Host:

1. In the artifact assembly diagram objects toolbar, click **Application Host** . 
2. Click in the diagram frame.
   A creation dialog box asks you to select a **Hosted Application**.
3. Select the application (in our example the electronic mail service application) and click **OK**.

Channels and communication ports

Communication channels support transfer of information between equipment resources. For more details on creation of these channels and the associated communication protocols, see "Communication channels", page 79.

Communication ports enable connection of artifact equipment resources with external equipment elements. For more details on how to connect communication ports between different components, see "Connecting Communication Element Points", page 80.

Describing Artifact Services and Requests

An equipment architecture is created to assure one or several services. For example, the software test platform is used to understand malfunctions and to carry out installation tests.

☛ *For more details, see "Service points", page 75 and "Request points", page 76.*

Describing Interactions in an Artifact Assembly Diagram

In an artifact assembly diagram, the *Interactions* enable representation of exchanges between equipment resources.

📖 *An interaction represents a contract established in a specific context between autonomous entities that are internal or external to an*

enterprise. These entities can be enterprise org-units, applications, activities or processes, as well as external org-units. The content of this contract is described by an exchange contract.

☛ *For more details on interactions, see "Managing Interactions", page 77.*

You can define interactions between:

- two equipment resources to represent terms of use of a resource by another in the context of the modeled resource architecture. For example, a business function application can depend on services provided by the electronic mail service.
- a service point and one or several artifact components to represent implementation of the service within the equipment architecture. For example, several software applications cover the required service.
- a component of artifact component type and a request point to represent that an equipment resource calls an external service.

DESCRIBING COMMUNICATIONS OF A COMPLEX INFRASTRUCTURE

In a complex infrastructure, communications are based on:

- service and request points,
- interactions,
- communication ports and channels.

HOPEX IT Architecture offers consistent and homogeneous features to manage each of these components and check relevance of the specified information.

Describing services and requests

Service points

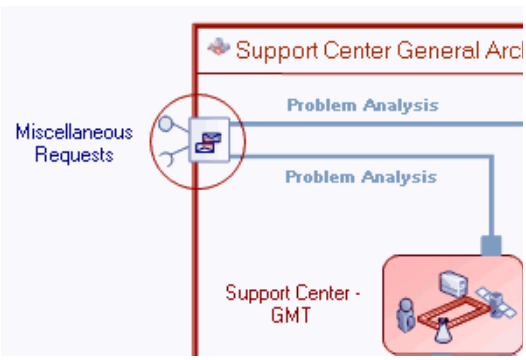
Services assured by resource architecture elements are represented by some *service points* .

A service point is a point from which the system receives a service request from another system, and provides the service requested.

The service is requested according to precise terms defined by an *exchange contract* assigned to the service point.

An exchange contract is a model of a contract between organizational entities. This contract is described by exchanges between an initiator role and one or several contributor roles.

Resources activated to assure a service are linked to the service point by interactions. If activation of several resources is necessary, then several interactions must be created between the service point and the architecture resources.



In the example here, the support service is linked to two interactions, based on the same contract, that represent

activation of one support center or another as a function of time slot.

To create a service point, see "Creating a Service Point or a Request Point", page 76.

Request points

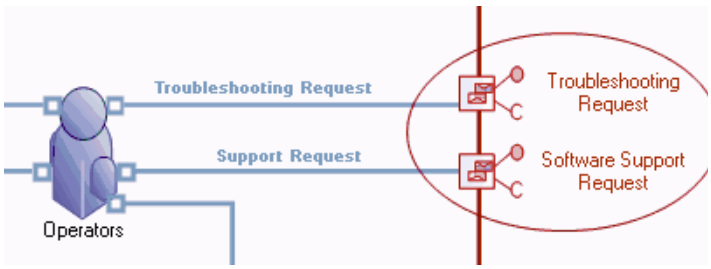
A *request point* enables representation of use of an external service.

A request point is a point from which the system sends a request to another system, and receives the service requested.

The service is requested according to precise terms defined by an *exchange contract* assigned to the request point.

An exchange contract is a model of a contract between organizational entities. This contract is described by exchanges between an initiator role and one or several contributor roles.

Resources that issue a request are linked to the request point by an interaction.



In the example, request points represent service requests made between call center operators and other organizations.

The request point creation procedure is identical to that for service points. For more details, see "Creating a Service Point or a Request Point", page 76.

Creating a Service Point or a Request Point

The process for creating a *service point* or *request point* is identical.

A service point is a point from which the system receives a service request from another system, and provides the service requested.

A request point is a point from which the system sends a request to another system, and receives the service requested.

To create a service point:

1. In the service architecture structure diagram insert toolbar, click **Service Point** .
2. Position the object at the edge of the architecture frame.
   A creation dialog box opens.
3. Click **Select Provided Service** to define the protocol that enables activation of this service point.
   A search dialog box opens.

4. Select the protocol that links this service point with the exterior.
   The role is automatically updated.
5. Click **OK**.

Managing Interactions

An *Interaction* represents the exchange of information between organizational entities of the system.

> 🕮 *An interaction represents a contract established in a specific context between autonomous entities that are internal or external to an enterprise. These entities can be enterprise org-units, applications, activities or processes, as well as external org-units. The content of this contract is described by an exchange contract.*

Content of an interaction is described by an *exchange contract*.

> 🕮 *An exchange contract is a model of a contract between organizational entities. This contract is described by exchanges between an initiator role and one or several contributor roles.*
>
> ☛ *For more details, see chapter "Describing Interactions", page 49.*

In the context of complex infrastructures, interactions are physically supported by communication channels. See "Communication channels", page 79.

Creating an Interaction

To create an interaction:

1. In the resource architecture diagram objects toolbar, click **Interaction** .
2. Draw a link between the two communication entities.
3. In the add interaction dialog box, specify the protocol you wish to use.
4. Click **OK**.

Connecting interaction points

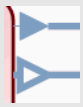
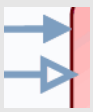
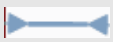
The interaction point connects an interaction to one of the components in communication. It enables specification of:

- the service point or query point that intervenes in the communication.
- the role, consumer or supplier, played by the interaction point in the exchange protocol.

To connect the interaction point:

1. Right-click the interaction beside the communication entity and select **Connect > Interaction Point**.
   A dialog box opens, presenting the list of service points and request points of the component in communication.
2. Select a service point or request point.
3. Click **OK**.

The graphic representation of the interaction point gives indications on nature and and consistency of information entered.

	The interaction point is not defined
	An interaction point has been defined, but no information has been indicated for the role played by the interaction point in the protocol
	The interaction point, defined or not, plays the role of consumer in the protocol
	The interaction point, defined or not, plays the role of supplier in the protocol
	There is a modeling error: the two interaction points play the role of supplier in the protocol
	There is a modeling error: the two interaction points play the role of supplier in the protocol
	Modeling of interaction points is correct

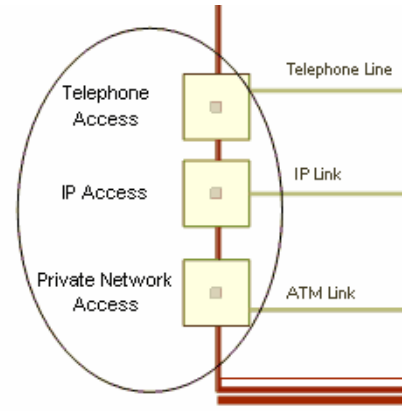
Describing Communications at Equipment Level

Communication ports

Communication Ports are physical points of communication that can be defined in artifacts and resource architectures.

> *A communication port is a physical point of communication with a resource. These conform to specific communication protocols. A communication port implements service and requests points.*

They assure physical transfer of information exchanged on service points and request points.



Communication ports comply with specific "Communication Protocols". See "Network communication protocols", page 79.

Communication channels

Communication Channels enable connection of equipment resources between themselves, with organizational resources or with communication points.

> 🕮 *A communication enables physical connection between two equipment resources. It supports interactions that define communication between these resources. Communication channels connect resources with the exterior via communication ports.*

Creating a communication channel

To create a communication channel:

1. In the resource architecture diagram objects toolbar, click **Communication Channel** .
2. Draw a link between the two communication entities.
   The channel appears directly in the diagram.

To define the communication protocol associated with the channel:

1. Right-click the channel and select **Connect > Communication Protocol**.
2. In the query dialog box that opens, double-click the required communication protocol.
   The protocol name appears alongside the channel name.

Network communication protocols

A *Communication Protocol* is supported by a communication channel.

> 🕮 *A communication protocol is a set of standardized rules for transmission of information (voice, data, images) on a communication channel. The different layers of protocols can handle detection and*

processing of errors, authentication of correspondents, management of routing.

For example, an SOAP protocol could be based on HTTP SMTP or FTP protocol for transport, themselves based on TCP, which is itself based on Ethernet.

A user may wish to build a customized layer of communication protocols and assign these to communication ports and communication channels.

☛ *Communication protocols supported by a communication port must be compatible with the communication ports to which they are connected.*

Connecting Communication Element Points

Like interactions, communication channels are represented by a link, extremities of which have graphical significance. For example, when the extremity of the channel displays an empty square, this indicates that the communication element point (located at the extremity of the channel) must be connected to the port of the connected object.

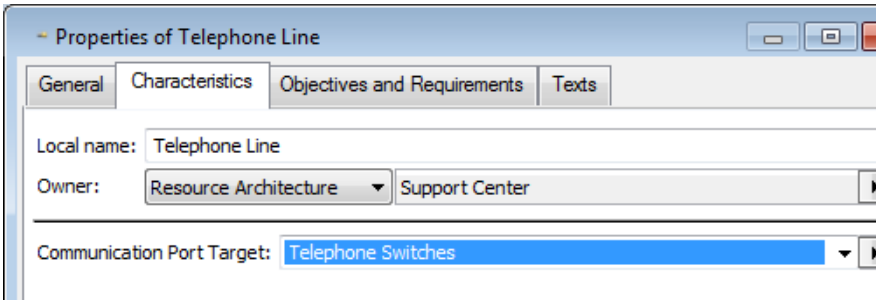
To connect a communication element point to the port of a physical asset or a resource architecture:

1. Righ-click the communication element point, at the extremity of the communication channel, and select **Connect > Communication Port**. A dialog box presenting the list of communication ports of the object opens.
2. Select a communication port.
3. Click **OK**.

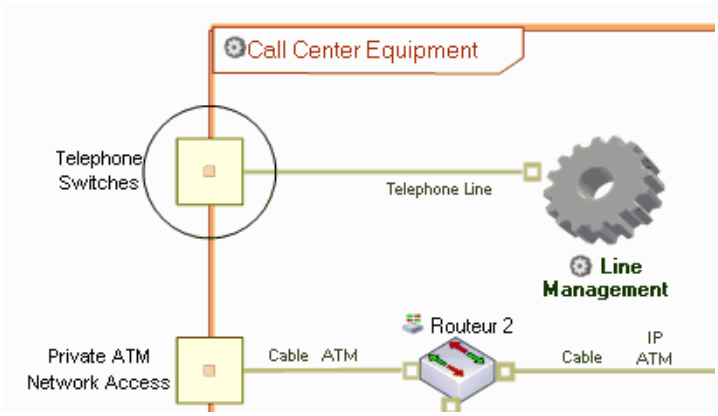
Example

For example, in the case of a support center, the "Telephone Line" communication channel connects the "Telephone Access" communication port to the general infrastructure of the support center.

From the properties dialog box of the communication element point, you can see that "Telephone Line" is connected to the artifact via the "Telephone Switch" communication port.



"Telephone Switch" is one of the ports available in the artifact.



SUMMARY OF TERMS USED

Diagrams and their objects

The table below summarizes terms used for the different elements of a complex infrastructure as a function of the context in which the component is implemented.

Diagram	Described element	Associated object	Comment
Resource architecture diagram	Physical asset	Artifact	
Resource architecture diagram	Human asset	Org Unit	
Resource architecture diagram	Architecture use	Architecture of resources	Use of an architecture by another one
Artifact assembly diagram	Artifact component	Artifact	Use of an artifact in another one
Artifact assembly diagram	Application host	Applications	The application is deployed with the object

Complex infrastructures are based on two types of concept:

- Elements described in the diagram, either equipment or organizational, that are owned by an architecture, such as: physical asset, architecture use, human asset or application host. Objects associated with these elements can be modified without impacting the elements themselves or their environment.
- Objects associated with these elements. These are resource architectures, artifacts, org-units or applications.

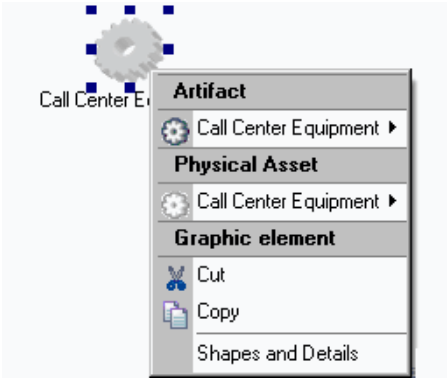
  In the example of support center general architecture, the GMT support center and the GMT+12 support center are uses of architecture. The "Support Center" describes the generic resource architecture implemented by both centers. This architecture is itself comprises different resources: physical assets and architecture use.

Objects pop-up menu

In a resource architecture diagram, the pop-up menu of an architecture element (for example a physical asset) presents commands specific to the object type deployed

for the element (artifact), followed by the commands relating to the physical asset itself. The third pop-up menu level concerns graphics.

☛ *If the type of element associated with the resource is not defined, the first part of the menu does not appear.*



Described element properties

To access the object that defines an element of a resource architecture:

1. Open the properties dialog box of the element (for example an architecture use) and select the **Characteristics** tab.
   The object that defines the element (in this case a resource architecture) appears in **Deployed Resource** field.
2. Click the arrow at the right of the **Deployed Resource** field to access the pop-up menu of the deployed resource or to replace it without changing the current element.

Associated object properties

You can specify properties on an artifact, a resource architecture, an org-unit or an application.

You can for example assign a transmission mode to call center telephones, a serial number or any other specific characteristic.

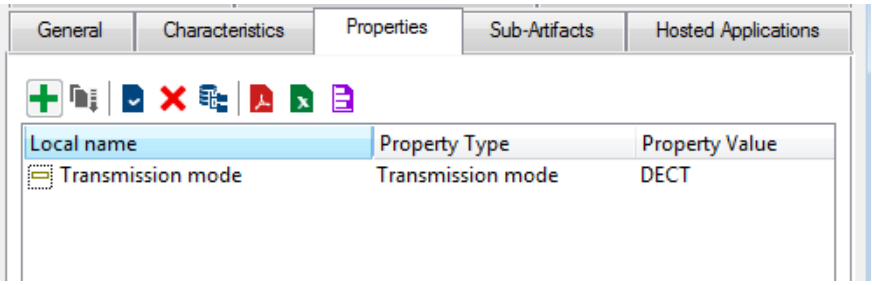
To define a property:

1. Open the properties dialog box of the object, select the **Properties** tab.

   ```
   Example: "Telephone" artifact.
   ```

2. Click **New** .
   The creation of property value dialog box opens.

3. Enter the name of the property, its type and value.



When the property type has been created, you can assign different values to it on different objects. Depending on objects therefore, "Transmission Mode" can take values "DECT", "Wi-Fi", etc.

To specify a new value:

- When adding a property to an object, using the arrow find the **Property Type** in question (here, Transmission Mode) and define a new value for the object to which the property applies, for example Wi-Fi.

Property inheritance

When an object inherits the properties of another, inheritance relates to the value of the property and not of the type. Therefore if you have defined on an object a property of "Transmission Mode" type with value "DECT", objects inheriting this object will all have transmission mode "DECT".

To modify the value of an inherited property, you must create a new property of the same type with a new value (for example, transmission mode "Wi-fi") and replace the inherited property by the new.

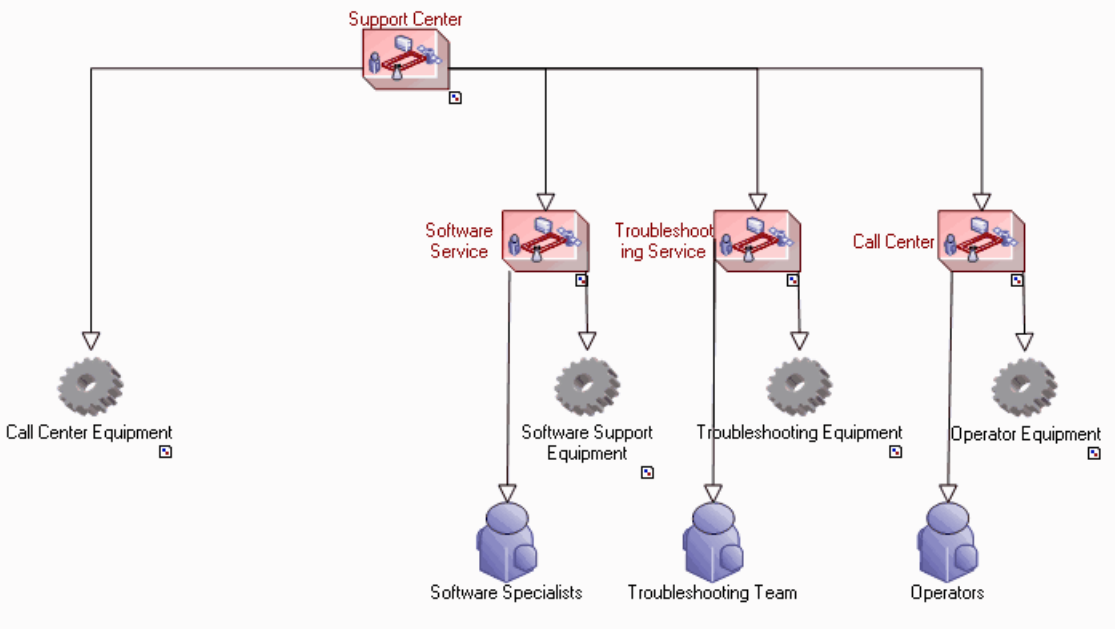
☛ *For more details on variations, see the **HOPEX Common Features** guide, "Handling Repository Objects", "Object Variations".*

Resource Architecture Tree

The resource architecture tree enables representation of resource types used in a resource architecture.

Resources represented are:

- human assets
- resource architectures
- artifacts



In the example of the support center, the resource architecture diagram presents equipment of the center and the resource architectures that enable request processing. Each of these architectures is itself comprises an org-unit and an equipment architecture.

APPLICATION DEPLOYMENT

HOPEX enables association of an application with one or several deployments. A deployment is supported by a site. On the same site, an application is deployed to offer different services to different users.

In addition, the problem of application deployment also consists of determining compliance between the infrastructure required to deploy an application and the existing infrastructure. The modeling tool must help in answering the following questions:

- What are the infrastructures required to deploy this application?
- Does the existing infrastructure enable deployment of this application?
- What are the missing elements?

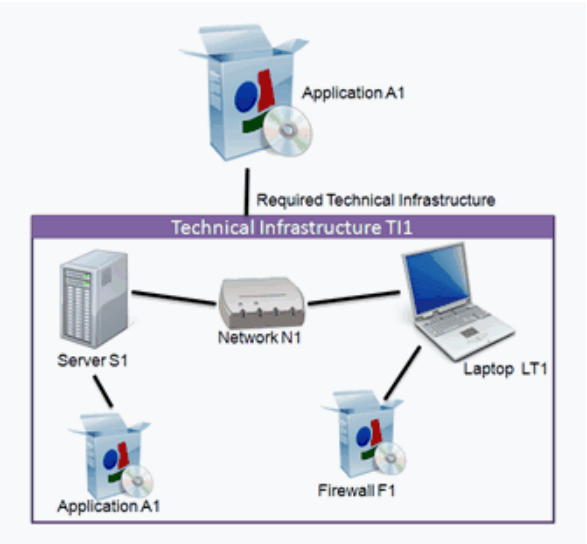
To answer these questions, two models are required: description of the infrastructure required on the one hand, and description of the existing infrastructure on the other. Mapping of these two infrastructures helps in determining the compliance level.

The following points are covered here:

DEFINING REQUIRED ARCHITECTURE

The following figure describes the technical infrastructure required for a given application. It also shows where the application is deployed in this infrastructure, and on which server it is hosted.

Specifying a required infrastructure not only enables listing of the elements required by an application, but can also position the application within this architecture.



☛ *To specify how an application should be deployed, you can use the concept of resource Architecture. To display objects linked to this model, you must select the Infrastructure Modeling Post 2009 SP2 option in **HOPEX** Business Process and Architecture Modeling options.*

To create the technical infrastructure of an application:

- Right-click the application and select **New** > **Technical Infrastructure**. You can then create the corresponding diagram.

See also:

✓ "Modeling technical infrastructures", page 43.

DEPLOYING REQUIRED INFRASTRUCTURE

So that an existing infrastructure complies with a required infrastructure, you must map elements of the two infrastructures.

The mapping editor offers mapping of the two models and allows you to create mapping between the elements of each.

An element of the required architecture can have a correspondence in several deployed architectures.

Creating a Mapping Tree

The mapping tree will allow you to deploy the required infrastructure of an application.

```
In the following example, you will deploy the
infrastructure of the "Office System" application in the
Bordeaux agency.
```

The use of the mapping editor varies slightly depending on whether you are in MEGA Windows Front-End or MEGA Web Front-End.

HOPEX Windows Front-End

To create a mapping tree:

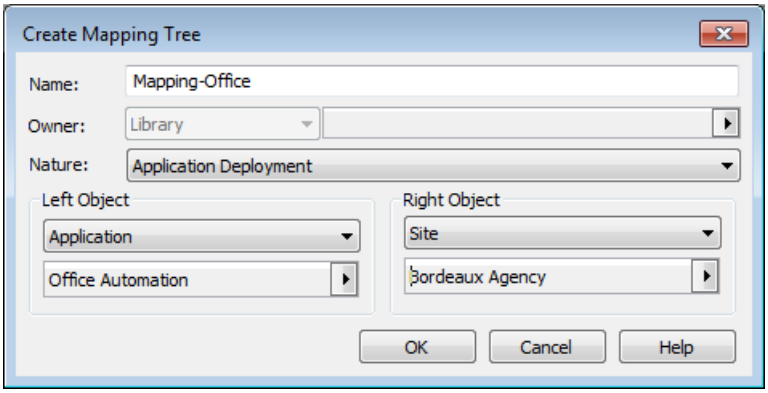
1. From menu bar, select Tools > Mapping Editor.
2. In the mapping editor menu, select **File** > **Create Mapping Tree**.
3. Indicate the name of the tree created.
4. In the **Nature** field, select the nature of the tree.

   ```
   Example: "Application Deployment".
   ```
5. In the **Left Object** pane, select the application to be deployed.

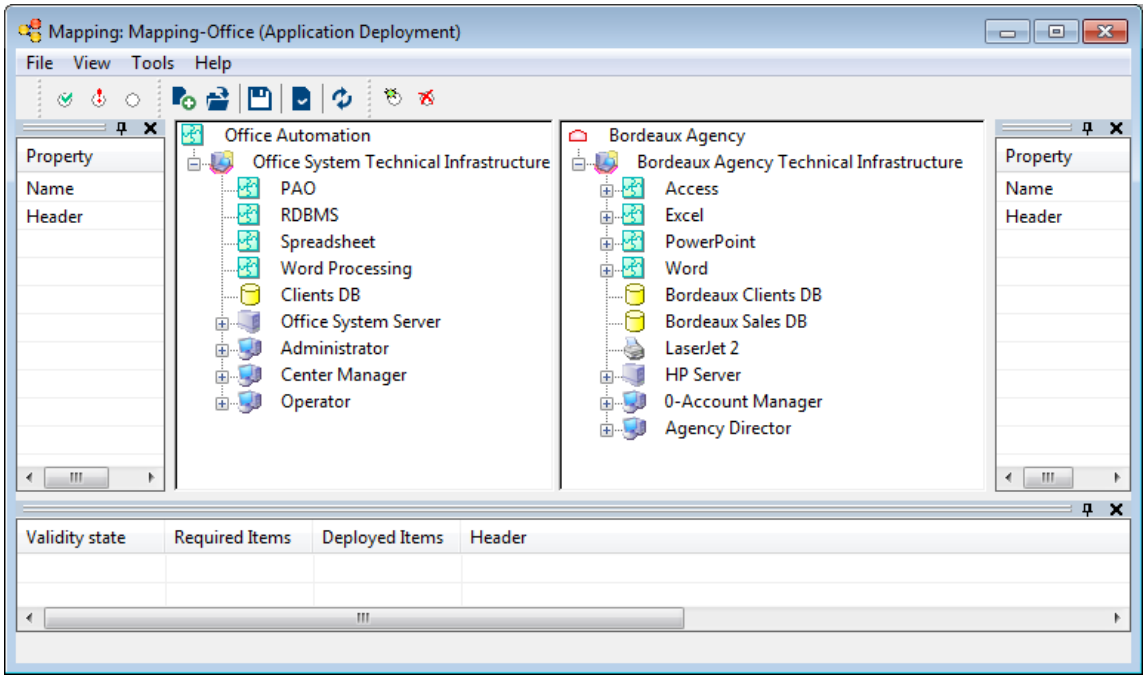
   ```
   Example: "Office system".
   ```

6. In the **Right Object** pane, select the site on which the application will be deployed.

   Example: "Bordeaux Agency".



7. Click **OK**.
   The mapping tree appears.
   The editor displays on the left the required technical infrastructure for the "Office System" application, and on the right the infrastructure for the "Bordeaux Agency".



8. Select **Tools** >**Mapping Editor**.
   A dialog box opens.
9. Leave the **Create Mapping Tree** option selected and click **Next**.
10. Indicate the name of the new mapping tree.

11. In the **Nature** list box, select the nature of the tree. In this case it consists of mapping data models.
12. In the **Left Object** and **Right Object** fields, from the object types concerned (data model), select the models you wish to align.
13. Click **OK**.
    The editor displays the mapping tree juxtaposing the two models.

HOPEX Web Front-End

To create a mapping tree:

1. Select **Tools** >**Mapping Editor**.
   A dialog box opens.
2. Leave the **Create Mapping Tree** option selected and click **Next**.
3. Indicate the name of the new mapping tree.
4. In the **Nature** list box, select the nature of the tree.

   ```
   Example: "Application Deployment".
   ```
5. In the **Left Object** pane, select the application to be deployed.

   ```
   Example: "Office system".
   ```
6. In the **Right Object** pane, select the site on which the application will be deployed.

   ```
   Example: "Bordeaux Agency".
   ```
7. Click **OK**.
   The editor displays the mapping tree juxtaposing the two models.

Establishing Mappings

To create mapping between an object of the required infrastructure (on the left of the editor) and an object of the existing infrastructure:

1. In the mapping editor, select an object in the infrastructure on the left.
2. Select the equivalent object in the infrastructure on the right.
3. Click the **Create mapping item** button.

   ☛ *In MEGA Windows Front-End, you can also create the mapping from the pop-up menu of the last object selected, by clicking **Map**.*

The mapping is created from the last object selected.

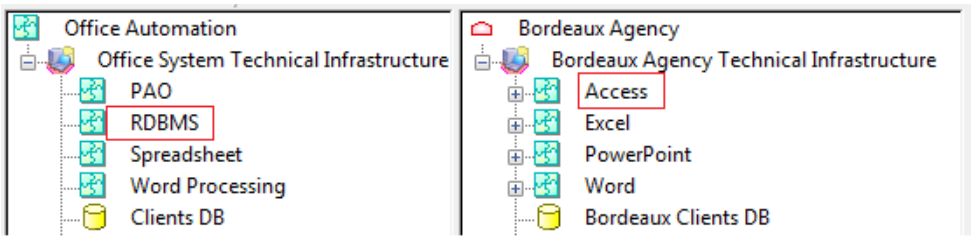
The mapped objects are marked with a green tick.

If a mapping cannot be created, an error message appears.

Example

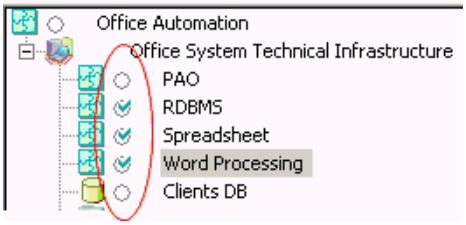
```
In the above example, to indicate that the RDBMS required
corresponds to Access in the agency, you must select the
```

RDBMS on the left, then that of the agency on the right, and create a mapping from the latter.



Object status

Indicators enable indication of status of synchronized objects.



A filter bar allows you to show all or only certain of these indicators. This bar is available by selecting **View** > **Toolbar** in the editor.

Object status can be characterized as:

- Valid
- Invalid (when an object has kept a mapping to an object that no longer exists)
- No mapping

ANALYZING DEPLOYMENT OF APPLICATIONS

Reports enable assessment of compliance level of an existing architecture with a required architecture model.

To check that deployment of an architecture complies with the required infrastructure:

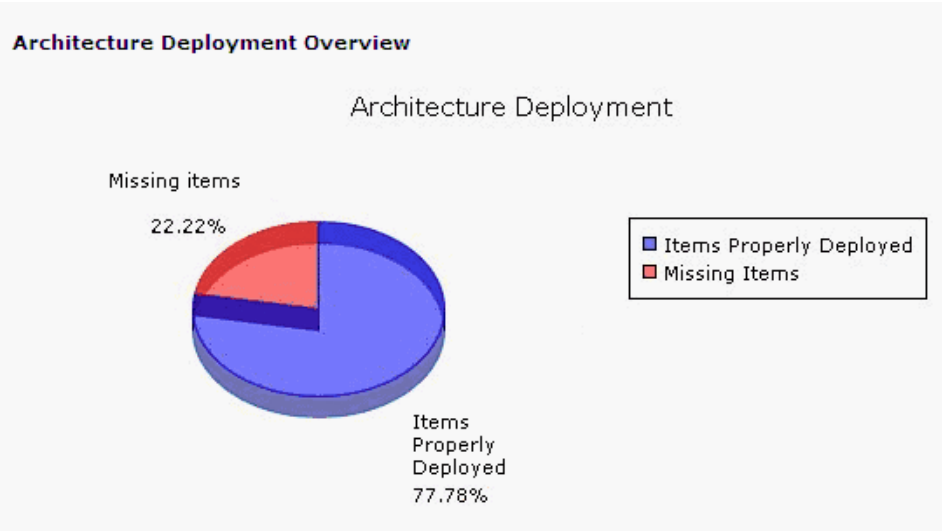
1. Right-click the application concerned and select **Report Discovery**.
   The list of reports available for the object appears.
2. Select **Architecture Deployment Compliance Analysis**.
3. In **Architecture Deployment Compliance Analysis**, click **Launch a New Report**.
   Three reports are available:
   - Architecture Deployment Compliance Overview
   - Architecture Deployment Compliance Details
   - Architecture Deployment Compliance Details Level 2

If no deployment has yet been carried out on the selected application, a message asks you to create one. In this case, you must select the application to be deployed, then the targeted architecture or site. See "Creating a Mapping Tree", page 89.

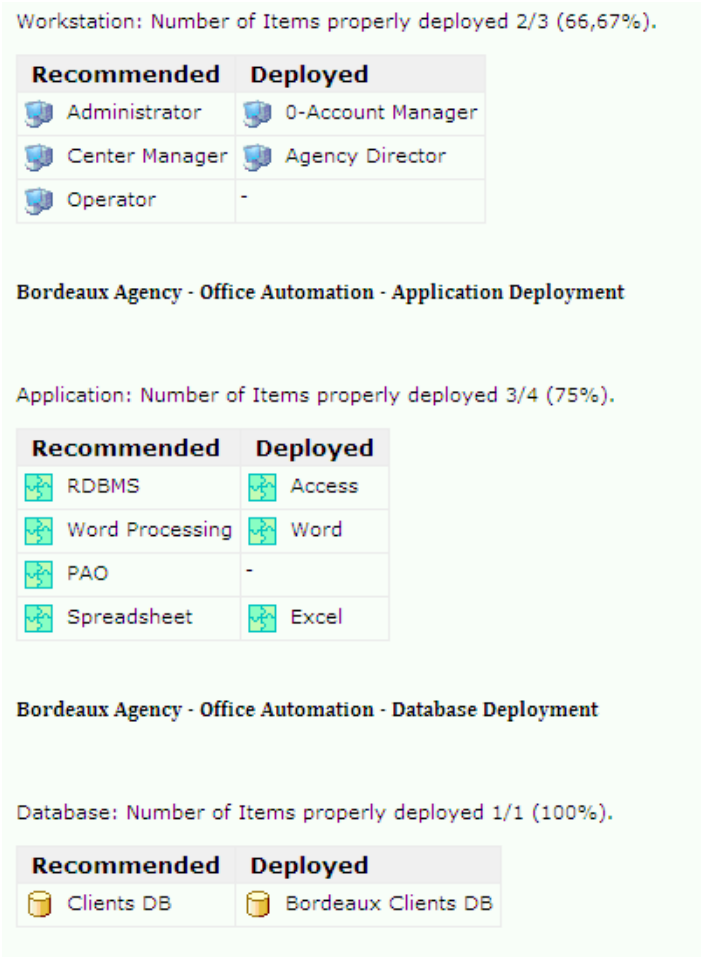
Analysis report example

In our example, we want to check the "Office System" application, common to the different agencies, has been correctly deployed in the Bordeaux agency.

After running compliance analysis, a first report chapter displays the global deployment result:



The last chapter displays more details; it gives the percentage of "Office Systems" infrastructure objects that have been correctly deployed in the Bordeaux agency and indicates missing objects.



Workstation: Number of Items properly deployed 2/3 (66,67%).

Recommended	Deployed
Administrator	0-Account Manager
Center Manager	Agency Director
Operator	-

Bordeaux Agency - Office Automation - Application Deployment

Application: Number of Items properly deployed 3/4 (75%).

Recommended	Deployed
RDBMS	Access
Word Processing	Word
PAO	-
Spreadsheet	Excel

Bordeaux Agency - Office Automation - Database Deployment

Database: Number of Items properly deployed 1/1 (100%).

Recommended	Deployed
Clients DB	Bordeaux Clients DB

Compliance indicators

Indicators of deployment quality are:

- the number of required infrastructure elements correctly deployed
- the percentage of objects correctly deployed
- the number of elements missing
- the percentage of elements missing.

INFORMATION SYSTEM CITY PLANNING

Enterprise information systems develop with wide differences over time to suit projects. Different types of applications are randomly accumulated from mergers/acquisitions or from consolidation/ deconsolidation. In large enterprises, there generally is significant redundancy in the data, services, and their use cases.

Under these conditions, it becomes increasingly difficult to integrate new applications into existing information systems and to handle technological evolutions.

The purpose of "city planning" for information systems is to define the main principles in the implementation and construction of computer applications, to ensure consistency throughout while decreasing the costs of building and integrating new applications. This should improve the company's ability to respond to a constantly changing environment.

This chapter explains how to describe the city planning for information systems using city plan diagrams.

The following points are covered here:

USE CONTEXT

City Planning Objectives

The main objective of city planning is to enable the information system to evolve progressively, without having to redesign the entire system, while allowing programs or software of different origins and periods to peacefully coexist.

For this to happen, it is necessary to define information system design rules that are valid for many years and are therefore independent of changes in technology.

City planning assumes that inventory of application assets has already been established, but it is not necessary to have documented the application architecture in detail; only the names of the applications and services is required.

A second objective of city planning is to identify functional redundancies in order to:

- Avoid creating new systems when developing new applications by using existing software resources.
- Reduce maintenance costs by carrying out renovations block by block, defining a new resource that is substituted for old ones and satisfies the various use cases.
- Consolidate two Information Systems in cases such as a merger/ acquisition.
- Prepare the deployment of EAI software on a large scale.

Note, however, that certain redundancies are justified. The fact that two use cases are similar does not necessarily imply that there must be a single solution. Other constraints (operation, performance, etc.) may lead to keeping them separate.

City planning justification

City planning is justified for businesses:

- that have significant application assets.
- that have a long history in information technology.
- where information technology is strategic.

It may also be justified in:

- Mergers/Acquisitions
  This immediately results in a high level of redundancy, particularly in operating resources. The city planning must occur prior to consolidating IT resources.
- Enterprises dependent on "software packages"
  Here again, the level of redundancy may be significant, particularly in the data. The objective of city planning is to prepare for data synchronization

Apart from these cases, city planning projects are rarely flagship projects. They must be low key, with no negative impact on other projects.

City Planning and Application Architecture

The application architecture defines or documents (maps) the structure of applications and their services, showing the cooperation between them. It explains how the information system functions.

The city planning for the information system explains what the applications ***do***, documenting redundancies so they can be reduced.

CITY PLAN

City planning consists of grouping the different applications used in the information system into "city planning blocks" of similar functionality.

Several grouping strategies can be defined. It is therefore possible to define several *city plans* describing the information system from different points of view.

> 🕮 *City planning is the splitting of the information system according to a particular criterion. This splitting can be by main enterprise functions, by origin of applications as the result of a merger, by type of system environment or by any other criterion pertinent to the context of the enterprise.*

Creating a City Plan

To create a city plan:

1. In the **Main Objects** navigation window, right-click the **City Plans** folder and select **New** > **City Plan**.
   The Creation of City Plan dialog box appears.
2. Enter its name.
3. Click **OK**.

The city plan is created and added to the list of city plans.

Opening a City Planning Diagram

To open a city planning diagram:

1. From the **Main Objects** Navigation Window, expand the **City Plans** folder.
2. Right-click the city plan concerned and select the diagram that interests you.

Duplicating a City Plan

A city plan can be duplicated with the aim of listing stages to be planned in its development.

> ☛ *HOPEXFor more details on duplication of objects, see the **HOPEX Common Features** guide, "Handling objects", "Duplicating objects".*

City plan names in a multilingual environment

When you duplicate a city plan, it must be in the repository language so that translations in the various languages will be correctly transferred to the duplicates.

☛ *To determine the language of your repository, consult repository properties.*

CITY PLANNING LEVEL

The information system processing is broken down into different levels:

- Area
  This is the highest level in the breakdown of the enterprise information system. For most companies, there is a data acquisition area, an information processing area, a management area, a repository area, etc.
- District
  The district groups the processing that can correspond to an activity or a business function, and therefore to a type of information.
- Block
  The block corresponds to the basic component of the city planning. A block is a homogeneous set of data and processing. Computer applications are grouped within the city planning blocks.
- etc.

☛ *Additional levels can be specified if required.*

Areas and Sub-Areas

"Sub-areas" can be created from an area, whatever the city planning level ("area", "district" or "block").

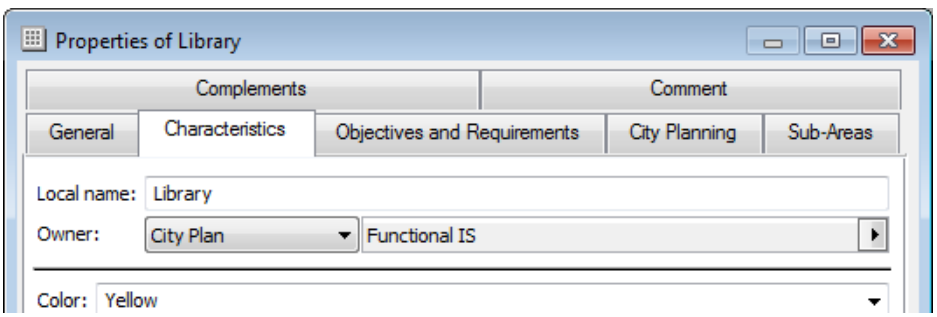
- In a city plan, when you create a **sub-area** from an "Area" (right-click the city planning area **New > City Planning Sub-Area**), the sub-area automatically becomes an area of "District" level.
- In a city plan, when you create a **Sub-area** from a "District", the sub-area automatically becomes a "Block".

☛ *The appearance of the area changes according its city planning level.*

Properties of areas, districts and blocks

To view the **City planning level** of the area:

❱ Open the properties dialog box of the area.



The **Color** attribute brings a further dimension to your city planning diagrams. The value given to this attribute changes the shape color representing the area, district or block displayed in the diagram.

The **City Planning** tab is used to specify the applications, databases, services, or use cases concerned.

Example of breakdown into areas, districts and blocks

For this purpose, we will use a fictitious enterprise called "Voyages & Vacations".

The Voyages & Vacations company was originally a sailboat manufacturer.

To sell and rent its sailboats, the company developed a network of offices and travel agencies in tourist destination islands such as the Caribbean Islands and Hawaii, as well as in major cities around the world. It also developed a network of banking offices to finance the purchase of its boats by private parties.

As technology changed, the original activity of boat construction declined, until only the activity of manufacturing spare parts for boats remained.

However, the boat rental and sales activity grew enormously. The company now offers pleasure boat rentals of all sorts, including both sailboats and motorized craft, and related services such as diving equipment rentals, car rentals, hotel reservations, and airline ticketing. It also now has several cruise ships and its own restaurant chain.

The company is organized into three main subsidiaries:

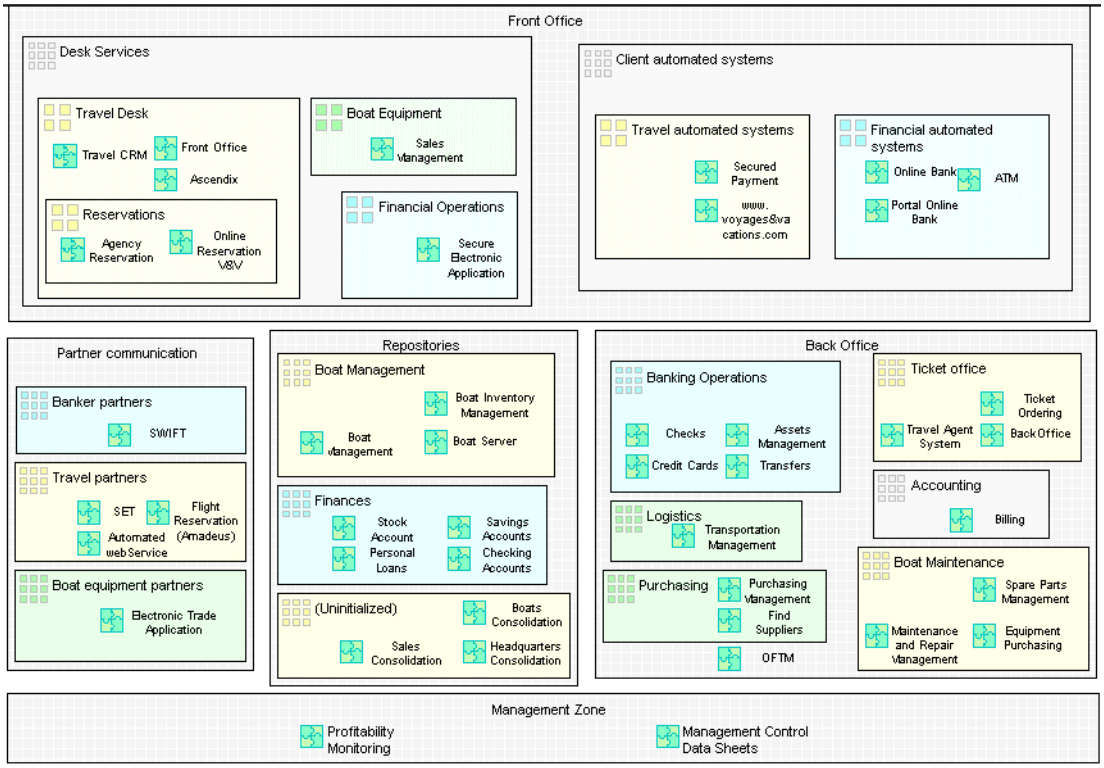
- The Voyages & Vacations company, which manages the rental of pleasure or cruise boats and related activities such as hotel reservations and airline ticketing.
- The MyBank company, where the banking activities of the group are handled.
- The HBC company, which manufactures and sells spare parts for boats.

Here is a detailed list of the various activities of the group:

- Boat rentals for boats of all sizes
- Airline ticketing and hotel reservations
- Car rentals
- Organizing cruises
- Organizing corporate seminars
- Managing technical documentation on boats
- Managing boat parts and boat repairs
- Purchasing boat parts
- Manufacturing and selling boat parts
- Managing libraries so books are available to passengers
- Managing the ATMs available to passengers on cruise ships
- Managing stock orders for passengers on cruise ships
- Other banking services
- Catering services on-premises or in the home

To contribute to the introduction of new applications into its information system, and to facilitate moving existing applications onto the Internet, the company IS manager began to map a city plan for his information system.

The following diagram shows a functional city planning diagram for this information system.



Functional city planning diagram for the information system

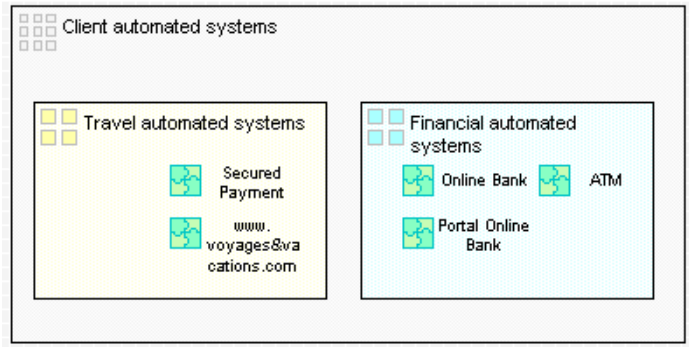
The information system is divided into areas, based on their interactions with the customers and partners of the company.

☛ *The color of the areas, districts, and blocks represents the subsidiary where these areas, districts, and blocks originate: Voyages & Vacations in yellow, HBC in green, and MyBank in blue.*

Data Acquisition Area

The first area, called the "Front Office", contains applications that are in direct contact with the customer. This area is divided into two districts:

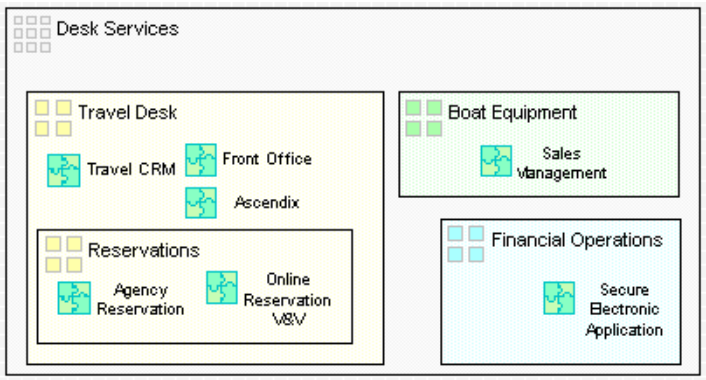
- "Client automated systems", which are automated systems used directly by the customer without any intervention by company personnel, such as Internet reservation systems or automated teller machines.



"Customer automated systems" district

This district is divided into two blocks, one containing the systems used for managing travel, the other the financial systems.

- "Desk services" district, which are services provided by people in the company, in the presence of the customer. These include travel reservations.

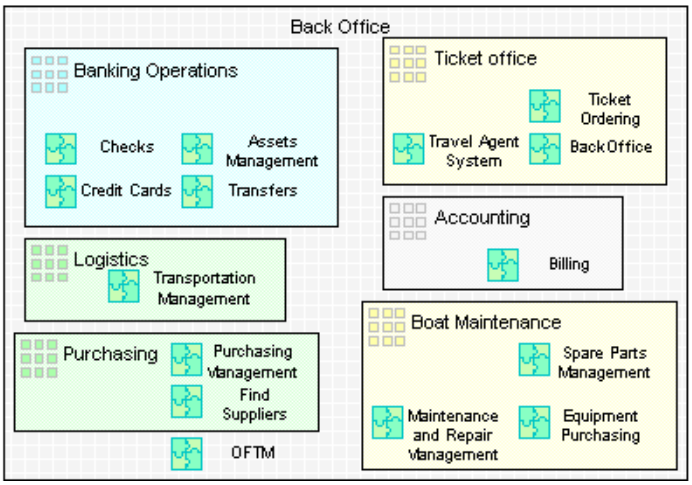


"Desk services" district

This district is divided into three blocks, one containing the applications used for travel management, one containing the financial operations, and a third the applications used for the sale of boat equipment. A specific sub-block has been identified for managing reservations.

Data Processing Area

A second area, called the "Back Office", contains the applications used for customer service that do not require the customer's presence.



"Back Office" area

This contains various districts, dedicated to "Banking operations", "Ticket office", "Logistics", "Purchasing", "Boat maintenance", and "Accounting".

Other areas

The "Repositories" area contains the applications used for accessing the persistent data of the enterprise.

The "Management" area contains the applications used by corporate management to direct the corporate strategy. These are generally statistics or performance indicators determined by compiling the persistent data in the enterprise repository.

It may also be useful to define an area for communication with business partners, particularly suppliers.

COMMUNICATION CHANNELS

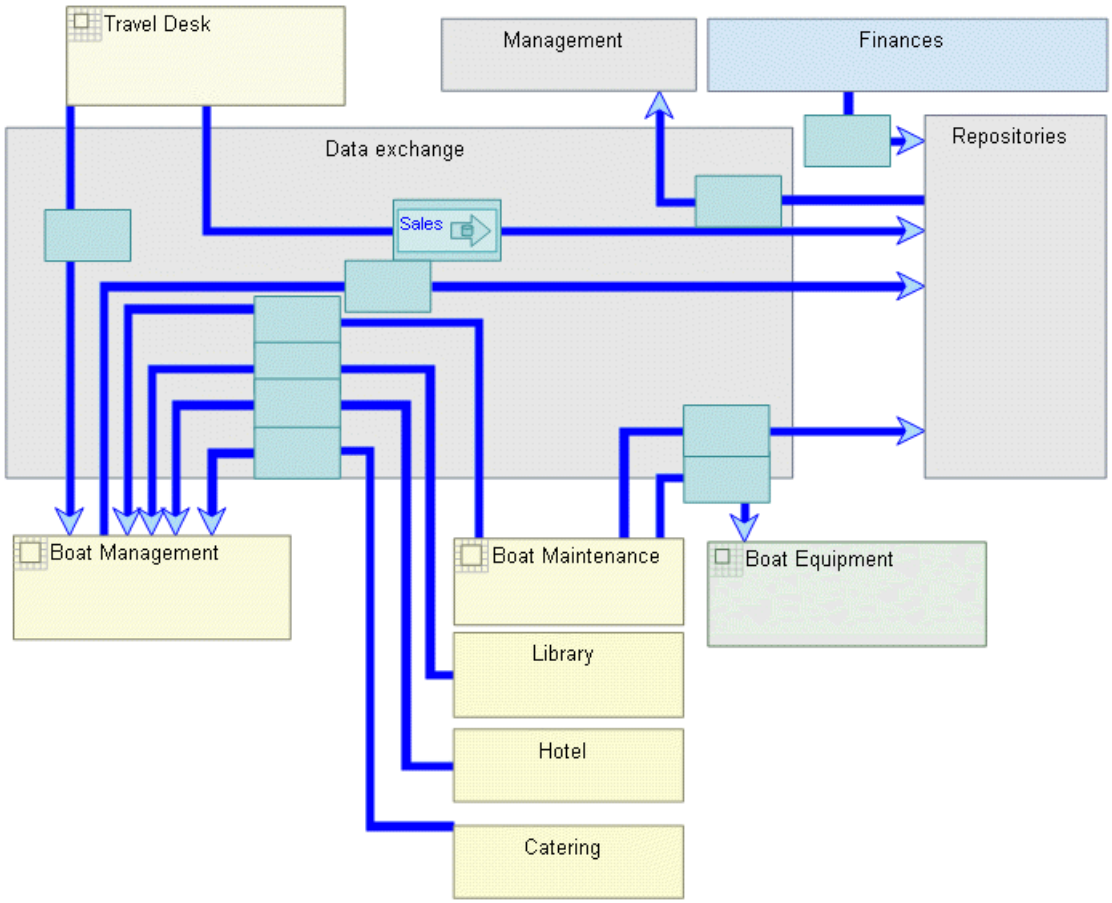
One of the main objectives of city plan is to facilitate changes to the information system by making it possible to modify an application block or add a new block while minimizing the impact on other blocks. For this to occur, it is essential that the communication *channels* between applications be identified, as well as the type of flows exchanged between them.

> 🕮 *A channel is a communication line through which the information flows (messages and their contents) transit between one sender and one receiver. The latter can be city planning areas, business functions, enterprise org-units, etc.*

The city planning diagram can show these communication channels.

Take the example of the "CPD Channels" city planning diagram which describes the "Functional IS" city plan.

This diagram shows the communication channels identified between the different areas and districts of the enterprise.



"Channels" city planning diagram

You can directly specify the content of the information carried by each channel in the **Exchanged Contents** tab of its properties dialog box. These contents correspond to information authorized to transit between two areas. Contents can be added in the channel direction (descending) or in the opposite direction. You can choose to use a single channel by positioning authorized contents in both directions, or to use two different channels for which only contents in descending direction are indicated. The latter solution is graphically clearer but creates more channels between the areas.

The contents indicated can be compared with those effectively carried by the applications in the source area and the channel target. This is achieved by compliance report. See "Application System Compliance with City Plan", page 110.

These contents can then be displayed in the diagram.

You can display the messages exchanged between the applications contained in each of the city planning areas, districts, or blocks connected by a communication channel.

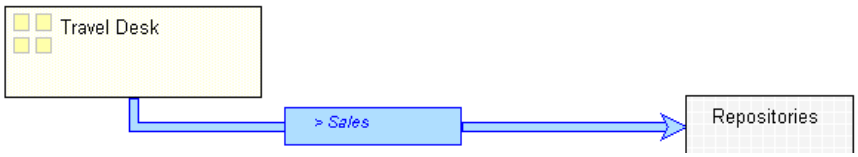
To display messages exchanged between applications:

1. Right-click the channel and select **Shapes and Details**.
2. In the **Display** dialog box, select the **Descending messages (Applications)** folder.
3. From the **Descending messages (Applications)** tab specify the messages exchanged by this communication channel that you want to appear in the diagram.

> ☛ *The messages listed are all the messages exchanged by the applications belonging to the areas, districts, or blocks communicating via this channel.*

You can repeat these steps for the ascending messages.

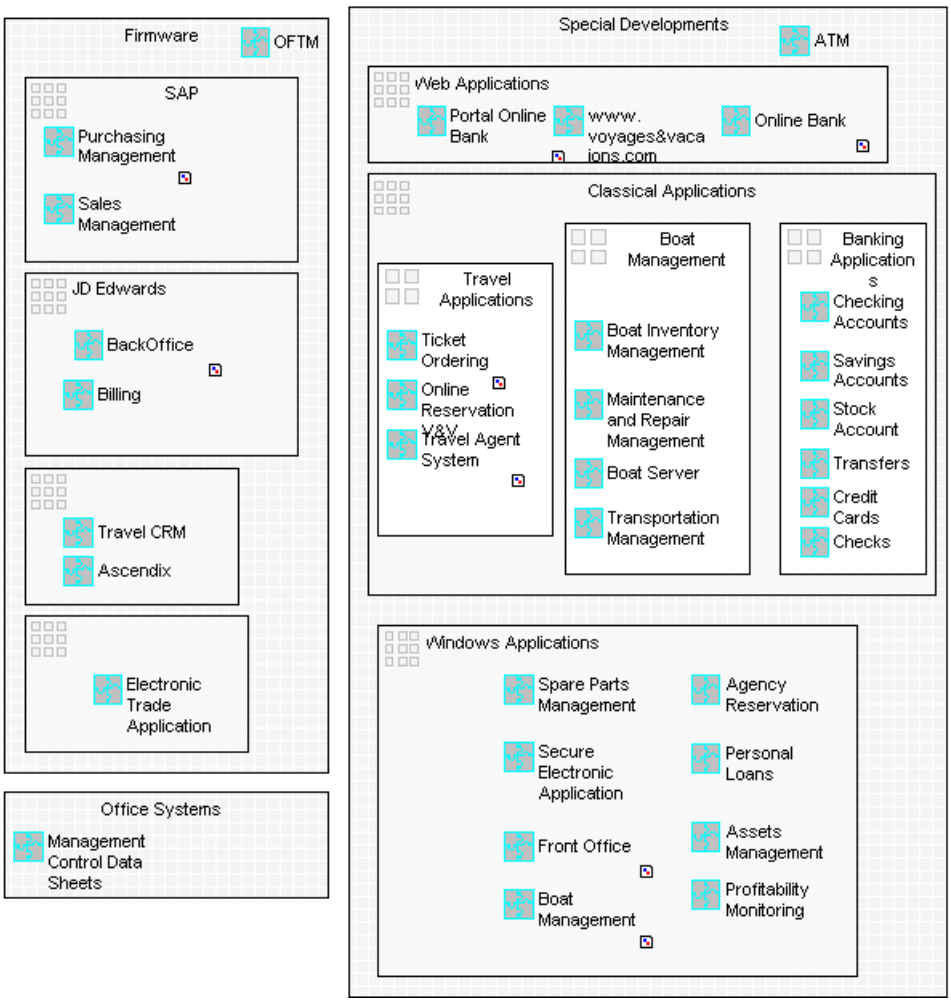
The selected messages are then displayed in the diagram, preceded by an arrow indicating the direction of the flow (ascending or descending).



CORRESPONDENCES BETWEEN CITY PLANNING AREAS

You can create several city plans describing the same information system, but based on different criteria.

The following diagram shows a breakdown of the enterprise information system based on purely technical criteria:

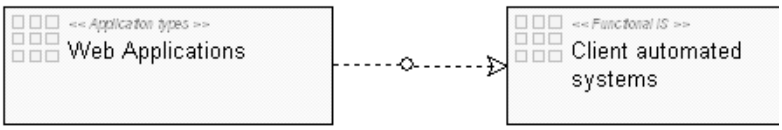


You can specify the *correspondences* between the areas defined in different city plans.

> 🕮 *A city planning correspondence enables specification of relationships that exist between areas, districts or blocks defined in the context of different city plannings.*

To do this:

1. From **HOPEX** menu bar, select **View > Views and Details** and check that the **Correspondences** view is selected.
2. In the diagram insert toolbar, click **City Planning Correspondence** .
3. Create the correspondence from one area to the other.
4. Indicate its name in the dialog box that appears.
   The correspondence then appears in the diagram:



You can connect this correspondence to other areas, districts, or blocks.

APPLICATION SYSTEM COMPLIANCE WITH CITY PLAN

Among the application architecture reports (see "Application Architecture Reports", page 46), **HOPEX IT Architecture** provides report template "IS Compliance with City Plan" which is a city plan check tool.

The purpose of this report template is to guarantee consistency of the system when adding applications.

Specifying City Plan Compliance Rules

An application system must follow rules imposed by the information system planner. These rules are defined at the level of each area and elements hosted in these areas must be compliant.

Two types of rules can be specified on a city planning area:

- **city planning area criteria**. These are HOPEX objects you can connect to the area to indicate that these objects must be implemented in this area. It can be:
  - an application
  - a functionality
  - a technical infrastructure
  - an artifact
  - an application service
  - a standard
- and **requirements**.

You can define these elements in the properties dialog box of a city planning area:

To indicate for example that an existing functionality must be implemented in a city planning area:

1. Open the properties dialog box of the area.
2. Select the **City Planning** tab.
3. Under the **City Planning Area Criteria** folder, right-click the **Functionality** folder and select **Connect**.
4. Find the functionality to connect to the area and click **OK**.

So that the application system will comply with the city plan, one of the hosted elements of the area must implement this functionality. See "Indicating Elements Hosted by the Area", page 111.

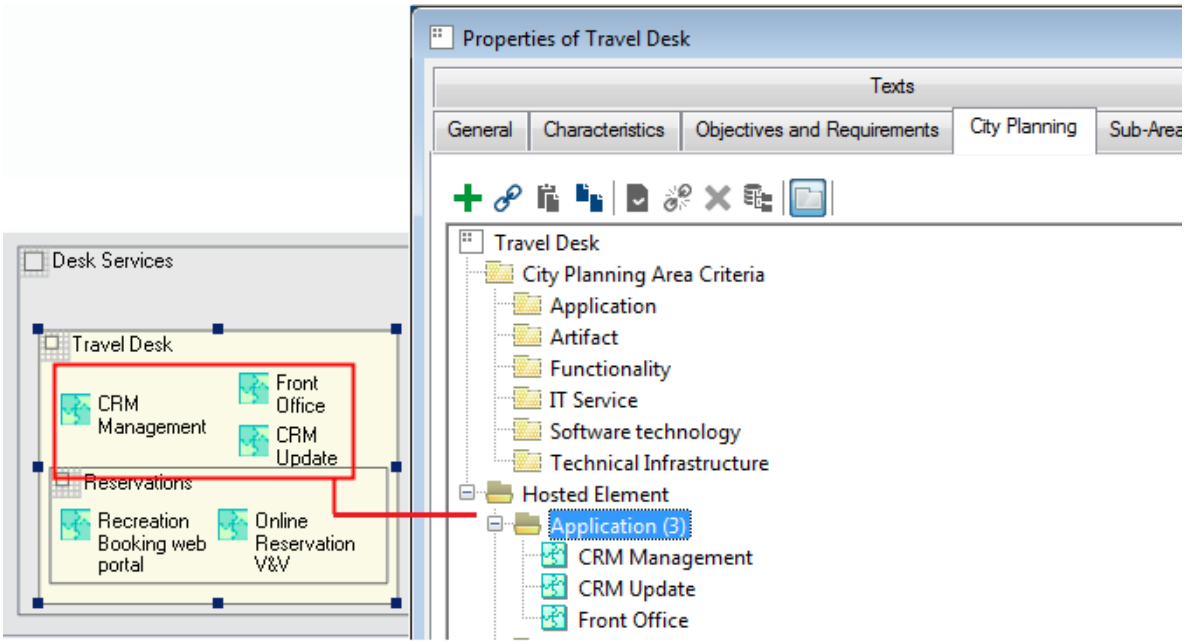
Requirements to be respected for the city planning area can be defined in the **Objectives and Requirements** tab.

☺ *You can analyze the hierarchical structure of a city plan. The report of the hierarchical structure enables identification of city plan area organization, and more particularly the definition of areas impacted by new functionalities.*

Indicating Elements Hosted by the Area

To determine best positioning of applications in the city plan, you can run a report "IS Compliance with City Plan". This places applications in appropriate areas according to the rules you have previously defined. See "Compliance Report Operation", page 112.

Depending on report results, or in the case of specific requirements, you can manually specify that a particular application is hosted by a particular area.



To indicate that an application is hosted in a specific city planning area:

1. Open the properties dialog box of the area.
2. Select the **City Planning** tab.
3. Expand the **Hosted Element** folder.
4. Right-click **Application** and select **Connect**.
   The query dialog box appears.
5. Select the application and click **OK**.
   The application appears in hosted elements.

Linking hosted elements to city planning rules

To be evaluated as complying with the city plan, elements hosted on an area must follow rules defined at the level of this area, in other words they must be linked to elements specified on the area (for example implemented functionalities).

Compliance Report Operation

The report template "IS Compliance with City Plan" relates area information with hosted elements so as to determine the compliance level of each application system.

The generated report can automatically position system applications in the different areas, ensuring maximum compliance level. This automatic positioning is based on comparison of applications with rules, but also takes account of interdependence of applications. Transfer of an application from one area to another can result in a reduction in compliance of another application (for example when transfer of an application disturbs exchange of data with another).

If the user has itself positioned the application in an area, the tool respects the decision of the user and will not reposition.

Evaluating IS Compliance with City Plan

To run a report on IS compliance with the city plan:

1. Select the **Documentation** navigation window.
2. Expand the **Report** folder.
3. Right-click the "IS Compliance with City Plan" report template and select **New** > **Report**.

   ☺ *Generally, you can display report templates available on an object (right-click the object and select "Report Discovery").*

   A wizard opens.
4. Indicate the report name and click **Next**.
   The wizard indicates the report template with its comment.
5. Click **Next**.
   You must then define the report parameters that correspond to the report input data.

   ```
   The "IS Compliance with City Plan" report template includes
   two parameters: IS Items and City Plans.
   ```

6. In the **IS Items** frame, click **Connect**  .
   A query dialog box opens.
7. Select the IS items to analyze, for example applications.
   The selected IS items appear in the wizard.
8. In the same way, in the **City Plans** frame, click **Connect**  and define the second parameter "City plan".
9. Click **Next**.
   You must then define the report chapters.
10. Select the report chapters.
    - City plan compliance: indicates applications that best correspond to city planning areas.
    - Report parameters: details input data and how it is interpreted in the different reports.
11. Click **OK** (MEGA Web Front-End) or **Finish**(MEGA Windows Front-End).
    The report appears in the list of reports.

To open a report:

- Right-click the desired report and select **Open**.
  Report results appear in the central page.

Report Results

The "City Planning Compliance" report chapter displays a list of compliance criteria:

- Functional completeness: checks that hosted elements implement all required functionalities. For a maximum score, no functionality must be excluded.
- Functional conformity: each hosted element must implement functionalities in its city planning area and in no other.
- Inter-area data transfer: communication channels carrying contents between city planning areas are sometimes defined. The analysis checks that these information flows comply with expected requirements between areas.
- Technological compliance: the hosted element must be implemented in an area complying with technologies defined in the city planning area.
- Requirements compliance: elements hosted in an area must be subject to the requirements of this area, in other words the element must be linked to these requirements.
- Structural compliance: the structure of elements should be respected; for example sub-applications of an application must be hosted in the same area as the application.
- Standards compliance: a compliant element is an element that respects the standards defined in an area.

Using these criteria, the report compares data defined on the city planning areas (functionalities, requirements, etc.) with those of the hosted elements.

> ☛ *These criteria are predefined, supplied by **HOPEX** to meet current requirements, but you can add your own report criteria. For more details on adding criteria, see technical article "City Planning Compliance - Adding an IT Criterion.pdf".*

A first table defines in which city planning areas the analyzed elements (in this case the applications named "Invoicing v1.0" and "Breakdown System v2.0") are best inserted.

You can see the results for each criterion by clicking the + sign alongside each element name.

The next table shows compliance levels of each hosted element (application), with the value obtained for each criterion.

Hosted Elements Levels of Conformity Table

	City Planning Area	86 % Global Conformity Level	80 % Functional Compliance	100 % Technology Compliance	100 % Structure Compliance
Invoice v1.0	Repositories	71 %	60 %	100 %	100 %
Breakdown System v2.0	Repositories	100 %	100 %	100 %	100 %

The final table shows compliance levels for each city planning area. In the example below, the "Repository" area displays functional completeness of only 71%, explained by the fact that three of the required functionalities required for this area are not implemented by applications in the area .

City Planning Areas Levels of Conformity Table

	71 % Functional Completeness
Repositories	71 % Those functionalities aren't implemented by any of the city planning area hosted elements: Display the Catalog Content Determine client situation

The report automatically places hosted applications in areas according to the elements that define it (eg. functionality, requirement, data flow, etc.). The choice is made so that the global compliance score is the maximum possible.

Certain applications may be followed by an icon representing a pushpin  This indicates that the application has been positioned in the area by the user, and not by the report.

> ☛ *Reports are dynamic; click* ***Refresh***  *(at top right of the report) to update the analysis to take account of latest modifications to the repository.*

SERVICE ORIENTED ARCHITECTURE (SOA)

Surfing on the technological wave of Web services, and largely responsible for the enthusiasm it generates, Service Oriented Architecture (SOA) is steadily progressing as an information system organization principle.

This new approach seeks to separate the service provided from the communication mode. Function calls are replaced by information exchanges carried by messages. With this service oriented approach, each service is able to manage its data and state.

The points that follow explain how to organize the service oriented architecture of your information system:

- ✓ "The Service, pivotal point between business functions and IT", page 118
- ✓ "Service Oriented Architecture principles", page 119
- ✓ "Adjusting the Application Architecture", page 125

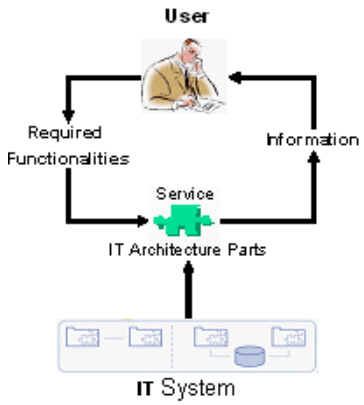
HOPEX System Oriented IT Architecture propose offers functions dedicated to representation and documentation of IT architectures using a service oriented approach. For more information, see the **HOPEX System Oriented IT Architecture** guide.

THE SERVICE, PIVOTAL POINT BETWEEN BUSINESS FUNCTIONS AND IT

With regard to the functional specifications, the business departments are able to express their needs – summarily – in the form of lists of information that they expect from the system. By doing this, they describe the features that ultimately need to be supported by a system.

As for IT teams, their task is to group as a service the messages delivering the requested information.
In an SOA context, a service is a consistent and indivisible processing unit that coordinates a set of messages and events, the objective being to execute one or several processing operations.

The services may be matched up to the operations executed at each step of enterprise business processes. Services can thus be seen as the pivotal point between the business functions and the information technology. The business functions perceive the services by way of the functionality provided, and the IT engineers perceive them as elements of the system architecture.



The service, pivotal point between business functions and IT

SERVICE ORIENTED ARCHITECTURE PRINCIPLES

Service Oriented Architecture is organization of the information system based on identification and arrangement of services and of the messages assuring their communication. The service itself is the cornerstone of the system architecture both in the specification phase and in the development and deployment phases.The SOA approach natively integrates principles of modularity, interfacing, exchanging by contract and interoperability.
The SOA approach natively integrates principles of modularity, interfacing, exchanging by contract and interoperability. In this way, it offers speedy adaptation of the IT system when faced with the development needs of the enterprise.

At the same time, it facilitates the leveraging of best practices thanks to the generation of reference patterns. These reference patterns may also be used for addressing the problems of systems convergence.

Designing a Service Oriented System

The service concept responds to four principles which are the foundation of information system design:

- Modularity: services that constitute a consistent arrangement of functionalities allowing the IT system to be designed as an assembly of autonomous, prefabricated elements.
- Interfacing: the service interaction boundary separating the data exchanged by the services from the internal processing carried out to produce these data. The data required and produced by a service define the service interaction interface.
- Exchanging by contract: the interface of a service is designed as a set of interactions between the service and its users. This reciprocity defines the roles of the service and of the users in the framework of an interaction contract.
- Interoperability: only the exchanged data are to be shared. So, for example, a COBOL program can be associated with a Java program once both are interacting as services exchanging messages. This involves switching from technical integration (COBOL / Java) to semantic integration (what is being talked about in the interaction?).
  Moreover, communication standards (HTTP, SOAP) and data description (XML) or interfacing (WSDL) standards have enabled access to information at a lower cost and for all types of organization, so multiplying the interoperability capabilities.

Designing the Reference Patterns

The advantages offered by the implementation of a service-oriented architecture include the possible uncoupling of these types of architecture from the applications

that implement them. This makes it possible to envisage the design of architectures dedicated to specific issues without the involvement of applications. In this type of architecture, the analyzed elements are pared down to the services, their interactions and the functionalities provided.

The design of these service architecture patterns may be motivated by convergence projects or projects for the implementation of best practices.

Service Oriented Architecture also finds other natural outlets such as component reuse, support service provision and capacity management improvement.

Reuse

Describing applications, right down to the details of the services that they comprise, enables identification of the elementary services that may be published for reuse when specifying a new application.

Support services provision

Service Architecture simplifies installation of support architectures dedicated to management of functional services: security management, communication management, performance management. Support architectures are key tools in information system governance.

Capacity management improvement

Service architecture simplifies administration and routing of messages enabling the implementation of capacity management. This means that a service which registers drops in performance when the load rate increases will be duplicated on several servers, with the service call message being routed to the server with the least load (Load Balancing principle).

Characterizing Services

Whether in designing a service-oriented system or in creating a reference pattern, service identification is a mandatory phase.

We can use the following typology to identify services:

- functional services
- services deduced from data models
- application services
- business process orchestration services
- technical and infrastructure services

These services can be deduced from analysis of enterprise business processes. Analysis of this business process results in a breakdown into operations executed by enterprise org-units (see the **HOPEX Business Process Analysis** guide for more details). For each operation, it is possible to determine to what extent the operation needs to be equipped with an IT service facility. This then brings to light a set of services that are used when the process is carried out.

Services deduced from data models

The data used in the context of enterprise processes also constitute indicators for service detection. In the above example, the operation: "Finalizing the credit proposal" illustrates the need to have one or more management services for the "Credit offering" entity. The credit is itself linked to the client entity whose properties need to be consulted.

The read, update, creation and deletion services for each of these entities can thus be defined. These four basic services are generally referenced by the acronym CRUD (Create, Retrieve, Update, Delete) and most often result in implementation in a relational database. From these core services are also derived the man-machine services that implement them.

Application services

Application services encapsulate algorithms for carrying out complex calculations or implementing business rules. In the credit example, a mortgage simulation service is used in the client interview. This service implements centralized computation algorithms and may be shared by other processes.

Business process orchestration services

Process orchestration services allow the chaining of the tasks implemented by the elementary services with the goal of automating all or part of the business process value chain.

For example, a flight booking system handles customer standby in the event of a flight being full. As soon as someone cancels, the service managing the process automatically handles the client call-back (sending an e-mail) and makes the booking. In this instance, the service manages the current status of the booking transaction (pending, reserved, canceled) for a period that may extend from several hours to several weeks.

Process execution engines available on the market (BPMS: Business Process Management System), are based on the formal description of the processes using a language. The most popular of these languages is BPEL (Business Process Execution Language). There is also a graphical notation to represent these business process definitions: BPMN (Business Process Modeling Notation).

See the **HOPEX System Blueprint** guide for more details.

Communication services

A Service Oriented Architecture is based on services and their interactions. Interaction management itself gives rise to the implementation of dedicated services. In an architecture of this type, it is essential to use protocols and standard services able to transmit messages between the various business function services. Since the services are of mixed type, the communication system is based on protocols that are independent of the messages transferred. It is therefore necessary to have services dedicated to "translation" of messages from their native format to independent format, and vice versa.

SOAP is a good example of a protocol used to address this issue. There are a large number of services that are capable of handling or transferring a SOAP message, whether this be on Unix, Windows or any other platform.

Administration services

The IT system is one of the main levers in process implementation. As such, it requires administration that allows decisions to be made in relation to its evolution. It is therefore a good idea to define a specific architecture for the administration of the IT system. The objective is generally to determine:

- if all the services implemented cater to requirements
- if they are correctly dimensioned for a performance viewpoint
- if they are continually accessible
- if they are regularly and easily updated, etc.

To address these internal issues, we may have recourse to the services defined below (this list is not exhaustive):

- Incident logging service;
- Network traffic analysis services
- Application update services
- Security administration services
- Material resource identification services, etc.

Security services

As for administration, the notion of security for the IT system may be considered beyond the functional implications. It is an entirely discrete issue which cuts across every service layer, whatever the business function involved Here, the objective is to protect the IT system against all kinds of attack, whether via software or hardware, and whether intentional or not.

Among the services corresponding to this problem we find:

- Access services: the objective is to control access to applications, servers or certain parts of the network. Firewall applications are dedicated to this problem.
- Authentication services: this involves checking the identity of persons trying to access software resources. Protocols such as SSO (Single Sign On) allow unified authentication for several applications used on the same workstation.
- Integrity services: the purpose is to guarantee that the transmitted data have not been altered during their transit. These services are generally based on calculating a key, whose value depends on the transferred content. On arrival, the key and content are compared to detect any integrity breach.
- Confidentiality services: this involves checking that the data transferred to one person may not be read by another. Encryption services are generally the solution here.

A number of organizations have taken on the duty of inventorying possible security risks in order to deal with them better. Among these we find ITIL (Information Technology Infrastructure Library) and CERT (Computer Emergency Response Team) recommendations .

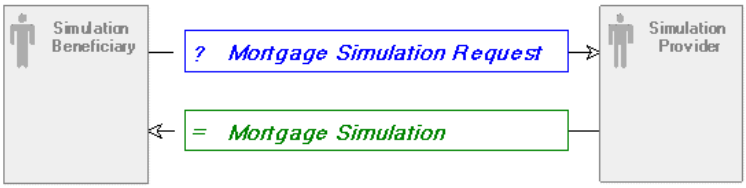
Identifying interactions

Interaction identification is carried out at the same time as service identification. Identifying a service allows you to identify the interactions that are associated with that service . For example, on the basis of a service called Credit simulator it is possible to identify an incoming message: "Request for credit simulation" and an outgoing message: "Credit simulation".

This example shows that one message alone is not enough to represent all the interactions necessary to obtain a result. To do this, it is necessary to group these messages within an interaction.

An interaction is an exchange contract between two or more partners. Each partner is represented by the role it plays in the exchange contract. Each elementary interaction in the protocol is specified by a message.

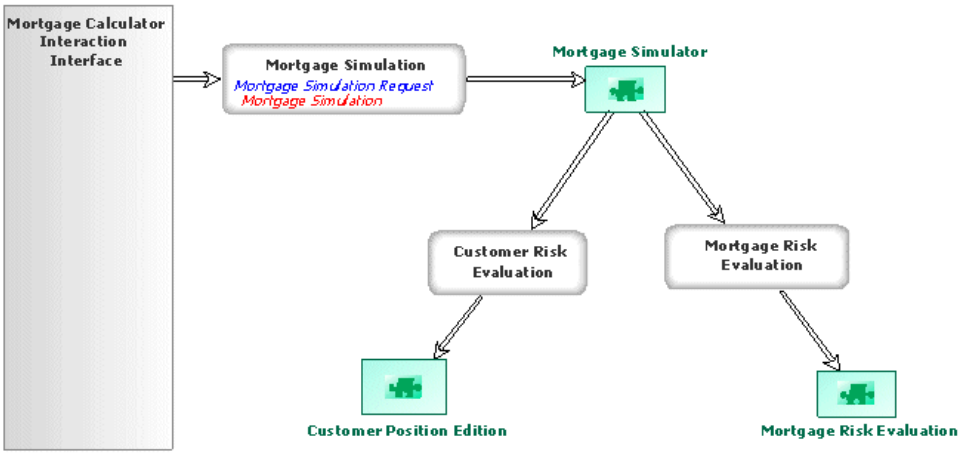
Interactions make it possible to describe the service interfaces, i.e.: all the input and output messages that they need to take into account. In our example, the Mortgage Simulation interaction comprises the two messages: Mortgage Simulation Request and Mortgage Simulation.



Interaction example

The "Mortgage Calculator" application offers a "Mortgage Simulator" capability. This plays the role of "Simulation Supplier". It must therefore reply to the "Credit Simulation Request" message with the "Credit simulation" message. The "Credit Simulation" internal application diagram shows how the "Credit Simulator" service is called via the "Credit Simulation" interaction. This same service must itself

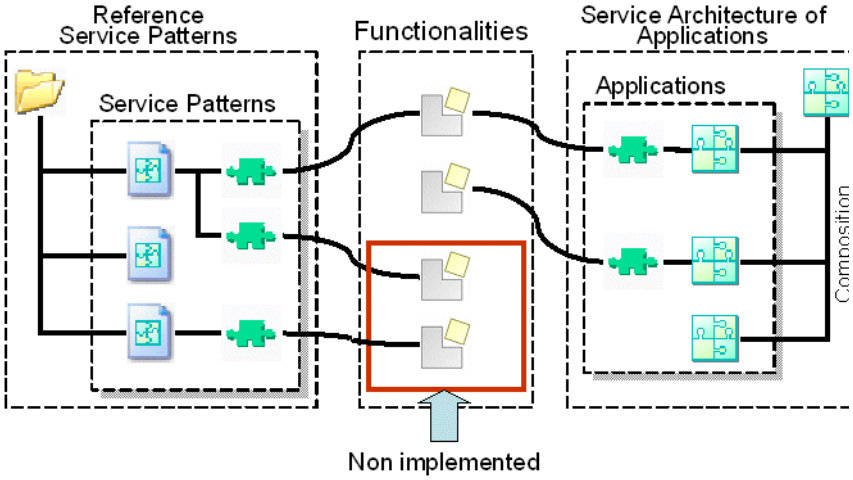
collaborate with the "Client Situation Consultation" and "Real Estate Risks Assessment" services to be able to carry out its processing.



Example of collaborating services – Credit simulation application

ADJUSTING THE APPLICATION ARCHITECTURE

Analysis of differences enables inventory of non-implemented functions.



Matching the application architecture to the architecture patterns

It is possible to evaluate:

- which features are not supplied by any application.
- how many existing applications cannot be integrated in the service architecture? (problem of interface, separation of services)
- for integrated applications, which services need to be reviewed. For example, because they have too many functionalities and therefore impose restrictions on the service architecture.

Integrating New Applications

When new applications are integrated into a set of applications, various criteria need to be taken into consideration, including:

- The durability of the application: a supplier study should be considered in order to predict the longevity of the product. We generally consider supplier-associated criteria such as maintenance and support conditions, customer references, the capital base of the company, etc.
- The technical constraints: even though the service architecture has been defined so as to abstract out the implementation constraints, there are often technical constraints that need to be taken into account from the start of the study. These constraints need to be considered when choosing the applications (and often it is better to bear them in mind when designing the service-oriented architecture in order to avoid embarking on studies when it is clear in advance that they are unrealizable). For example, if the operating system is an imposed choice

(on account of the existing hardware pool), the catalog of applications will inevitably be limited.
- The use of standards: consider whether the chosen application is based on standards, perhaps in terms of data export (XML format), data storage (relational base), the man-machine interface (reduced learning time), etc. These considerations determine the potential for future evolution.

Service Accessibility

Applications make available some of their services for other applications with a view to their reuse. To do this, they need connection points with the outside. Only the services linked to a connection point are accessible for a given application.

Deployment

this consists of defining the infrastructure on which are based the applications chosen for implementing the service-oriented architecture.

If the technical constraints have not been imposed at the outset, the choice of applications will certainly bring new constraints. For example, the operating system, the size of the host servers and the power of these servers are all examples of the parameters to be considered when implementing the technical infrastructure.

SOA Analysis Reports

HOPEX IT Architecture provides reports enabling service-oriented architecture analysis:

- Application Functional Analysis: this report template compares a set of elements providing functions with a set of expected functions (functional scope).
- IT Services Attached To Process: this report finds services relating to a set of processes.

See "Application Architecture Reports", page 46.

DETAIL OF CONCEPTS USED

This chapter presents details of the concepts used in **HOPEX IT Architecture**.

The diagrams used in **HOPEX IT Architecture** help you to design the architecture of your information systems.

Depending on your goals, modeling can provide you with a global approach for your entire information system, or for one of its sub-systems.

Similarly, modeling it can give you a software-oriented view, focusing on the applications, flows of information, and structures used to save data, or a hardware-oriented view that highlights the technical elements.

To address these requirements, **HOPEX IT Architecture** offers three types of diagram:

- Application Architecture Diagram for a software-oriented view of the information system.
- Technical Infrastructure Diagram, for a technical and hardware-oriented view of the information system.
- Applications Tree to indicate the hierarchy of the software components of an Application.

The following points are covered here:

APPLICATION ARCHITECTURE DIAGRAM

Application architecture diagrams enable description of the software environment of an *application*, an application system or of the enterprise itself. They also enable description of the internal architecture of an application system, an application or a *service*.

To respond to these different requirements, there are several specializations:

- Applications overview
  This diagram presents the main applications of the enterprise and their interactions.
- Application environment diagram
  This diagram presents exchanges between the application and its main user applications. It enables placing the application in its environment without considering its internal architecture. It defines exchanges expected or identified (expected interfacing contracts, "service contracts"). It does not however describe breakdown of the application into application modules, services, etc. and does not contain the applications, services or databases required for its operation.
- Application internal architecture diagram
  This diagram details exchanges between components (applications and services) involved in application behavior. It enables definition of architecture and internal exchanges required for correct application operation and the explanation of how exchanges with the application environment are processed.
- Service architecture diagram
  This diagram details exchanges between components involved in service behavior.

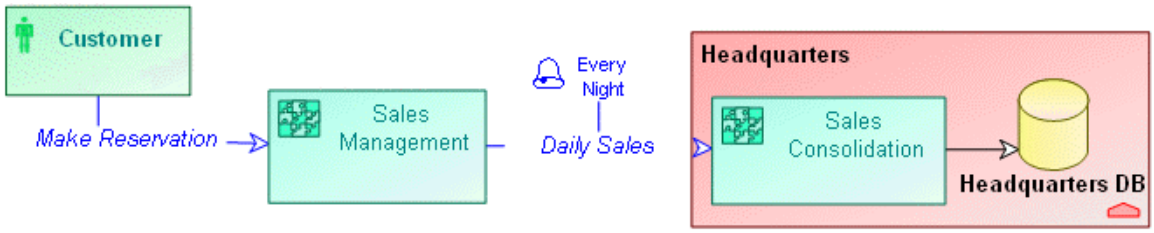
Its main concepts are applications and communications that exist between them.

Basic AAD Concepts

Within described scope, the AAD shows:

- the described *application* or the main applications of which interdependencies are being studied.
- the *Information flows* exchanged by the applications and any associated timers.
- the *sites* where applications run.
  An application can involve several sites. For example, a client-server application can run distributed processes, and an application can also be installed at several enterprise sites. Note that an application can be split into sub-applications.
- the *databases* consulted or updated by applications. Note that the sites where these databases are installed and the machines that use them are indicated.

- the *org-units* outside the organization (such as customers and suppliers) that send or receive messages (information flows) to and from the applications.

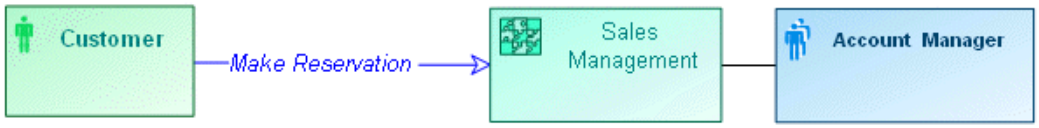


- the interfaces between the applications, the information flows, and the consulted and updated databases.
- information exchanged between databases if these exchanges (file transfers, database replications) are performed by small applications.

- the *services* used by applications to perform activities such as transferring data from one database to another.

Organizational elements: the org-units

The AAD can also include organizational elements such as the external and internal org-units that use applications, and the messages they send or receive.

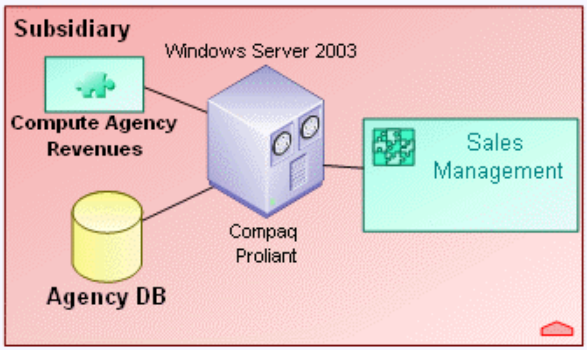


Hardware and technical elements: servers and workstations

The AAD can be enhanced with technical elements that enable adequate dimensioning of machines and networks.

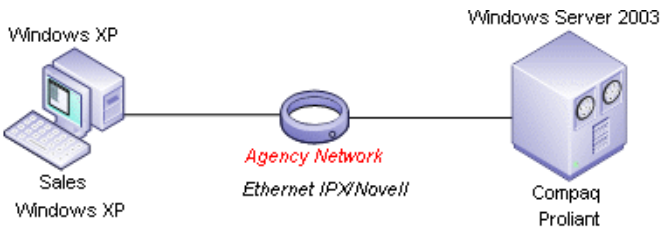
You can indicate the server where an application or a service runs, the geographical location of the site where this server is installed and the database(s) found on this server. When you know which applications run on a server, you are better able to

choose the correct configuration for this server and decide what type of computer you need, as well as its operating system, memory and hard disk capacity.



When you define the workstations, you are describing the computers assigned to the org-units. You can indicate which applications and databases are to be run on each workstation. This information allows you to make the correct choice in workstation configurations.

You can also specify the network that connects these different computers.

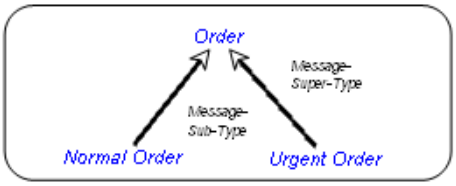


Specialization

Specializations allow you to distinguish between particular cases of the same message. Certain operations are valid for the general case known as super-type: for example, all the orders sent by a customer. Other operations (sub-types) will be

used only in specific situations. For example, orders placed using e-mail are processed differently from orders placed by fax.

```
An order placed by a customer is processed differently
depending on whether it is urgent or not.
```

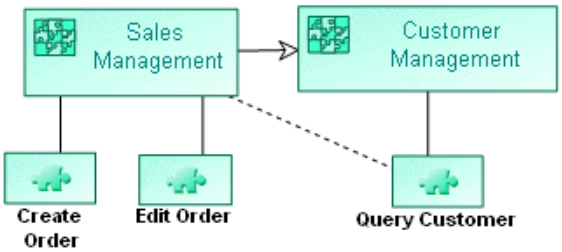


Specialization of messages

You can indicate sub-types for org-units, sites and messages.

Distributed processing

You can distribute processing between a client application and a server application. You can also indicate which services on the application server are used by the client.



Distributed processing

Inter-site Analysis

These are the links required or already existing between the sites.

Main Uses of the AAD

Depending on the scope of your analysis, you can choose to build an overview (global AAM of your organization) of the architecture of your information system, or focus on a specific sub-system to detail the basic building blocks (AAM of a particular Application and its environment).

The global approach consists of mapping the applications and positioning each of these within the general information system architecture. This approach defines the main sites, information flows, and data.

The detailed approach focuses on a software-oriented view of the information system. This allows you to study and represent how to split an application into

deliverable or substitutable parts. This approach often reveals the main org-units involved and, if required, the networks and hardware required.

TECHNICAL INFRASTRUCTURE DIAGRAM

The Technical Infrastructure Diagram (TID) focuses on software architecture strategic objectives.

This type of diagram (Technical Infrastructure Diagram or TID) provides a detailed description of the technical architecture: network management, information flows, routing.

The TAD also allows you to choose the correct hardware configuration for your applications and databases, plus the networks used to exchange information.

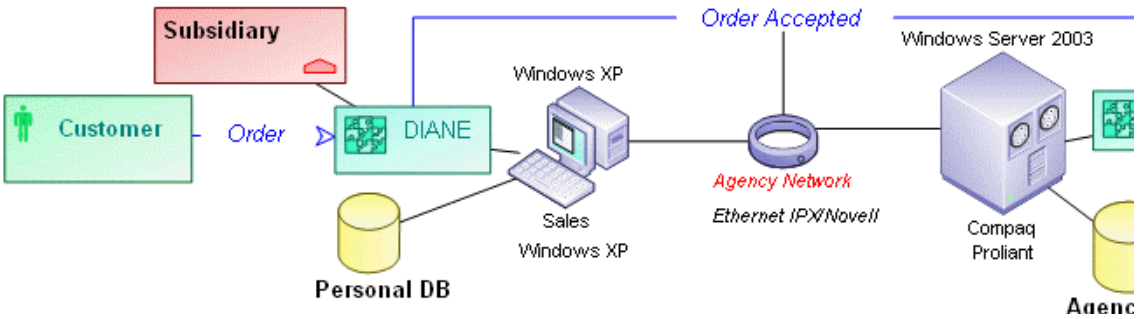
TADs focus on describing the networks and servers.

A technical infrastructure diagram can be used to define a "Technical infrastructure overview" of the enterprise, or to create the specific "Technical infrastructure diagram".

Basic TID Concepts

Within the described scope, the TID shows:

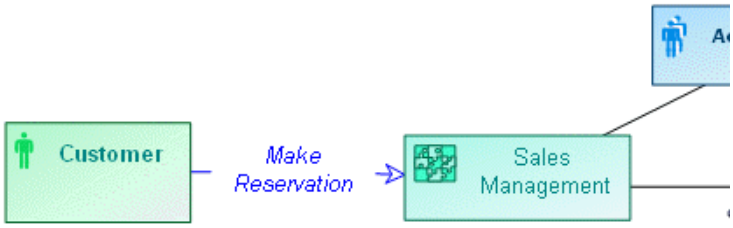
- the *networks* used to send information and connect client workstations to servers.
- for each site, the *servers* where the applications and databases are installed.
- the *workstations* and their installed applications and databases.
- the network *nodes* such as printers, modems, etc. These nodes can be connected to each other or to servers.
- the sites where networks, servers, workstations and other types of hardware are installed.
- the information flows exchanged by applications, databases, and external org-units. These messages can be sent across a network.
- the external *org-units* that send or receive messages used by applications.



Organizational elements: org-units, etc.

A TID can be enhanced by adding organizational elements.

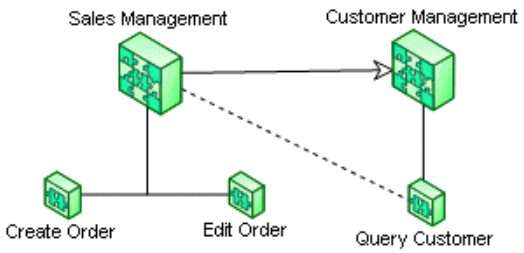
- For example, you can indicate the org-units that use an application across the network. This application must be accessible from the org-unit workstation.



- If a message describes a file transfer, you can use a timer to indicate when to transfer this file.

Distributed processing

You can distribute processing between a client application and a server application. You can indicate which services on the application server are used by the client.



Specialization

You can indicate sub-types for org-units, sites and messages.

Main Uses of the TID

Depending on the scope of your analysis, you can choose to build an overview of the information system architecture, or you can detail a part of it: for example, in a general overview you can indicate the methods of communication used between the different sites in your organization; in a detailed view, you can focus on one specific site to describe the technical infrastructure of this part of the information system.

The global view of an information system maps the geographical locations of the organization and shows communication between sites. This approach highlights the networks and servers used, as well as the applications and databases that send or receive information.

The detailed approach focuses on describing a specific site, in order to analyze and represent the technical infrastructure of the local information system and its main links with its environment. This type of approach highlights local area networks, the servers where the applications and databases are located, the connected workstations, and the peripherals used (printers, modems, etc.).

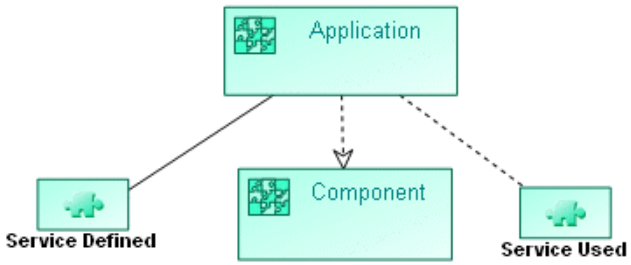
The detailed approach also allows you to describe a specific local area network, all the connected hardware, and all installed applications and databases.

APPLICATIONS TREE

The MEGA Architecture Application Tree (AT) is a **HOPEX IT Architecture** diagram that allows you to describe the software architecture of a particular application. Focusing on a particular *application*, this type of diagram enables representation of its division into modules (sub-applications) and basic services.

The AT contains:

- the main application of which components are being analyzed.
- the application modules (sub-applications) that make up the main application.
- services defined in this application, and services used by it but managed by other applications.



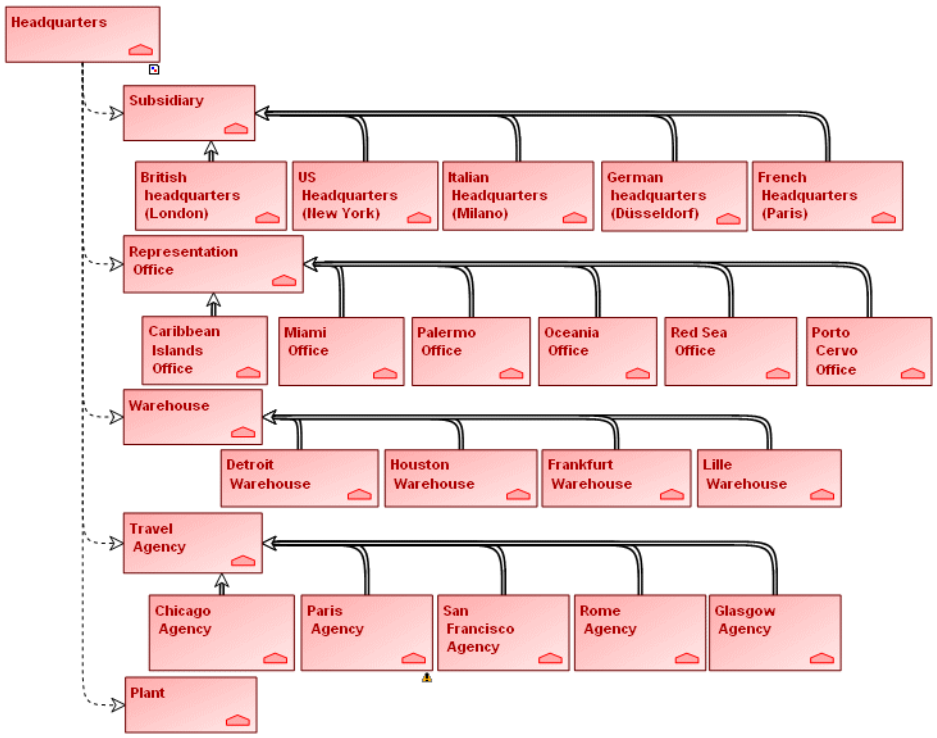
Application Tree

SITE TREE

The site tree is a **HOPEX IT Architecture** diagram that enables tree description of a site. Focusing on a particular *site*, this type of diagram enables summary representation of site breakdown into sub-sites.

The ST contains:

- the main site of which breakdown is being analyzed.
- sub-sites that make up the main site.
- geographical specializations of each site type.



Site tree

OVERVIEW OF SITES AND SITE FLOW ASSESSMENT

The overview of sites presents the assessment of flows between the main enterprise sites.

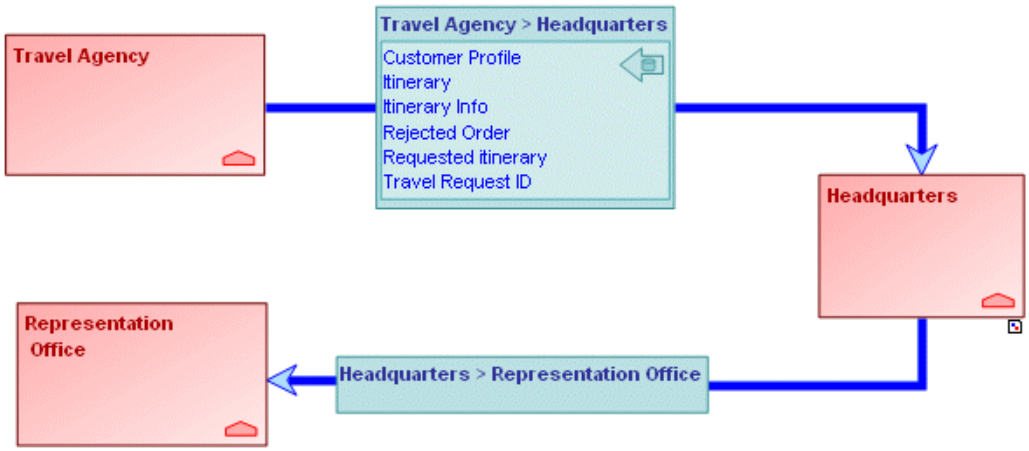
For a given site, assessment of flows between sites presents assessment of exchanges between its components and/or with other sites.

This assessment is produced using channels that count flows exchanged between elements (applications, services, etc.) hosted by each site.

This diagram shows:

- sites that exchange data.
- exchange channels between these sites.

The channels are automatically completed with contents of messages exchanged between applications, services, etc. hosted by each of the sites.



Assessment of flows between sites

CONCEPTS MANAGED BY HOPEX IT ARCHITECTURE

The sections that follow explain exactly how to use each object managed by **HOPEX IT Architecture**.

They highlight the various links you can use to connect different objects in diagrams.

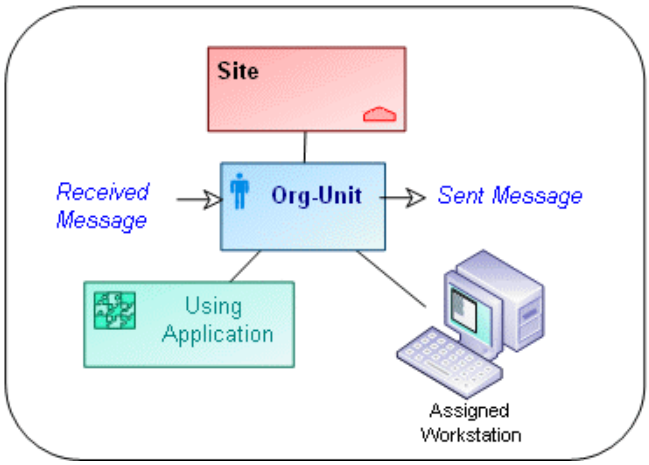
As explained above, certain objects can be used only in one diagram type: for example, nodes can be used in a TID only.

In addition, certain objects can only be accessed in a diagram if you enable the corresponding view focusing on a software, organizational, conceptual or technical view of the information system.

HOPEX IT Architecture can manage an object hierarchy containing objects of the same type. This means it can handle the following concepts:

- **Composition**: an object is defined as the aggregation of several other objects of the same type but of reduced scope. For example, the "Agency Reservation" application consists of N sub-applications (x, y, z, ...).
- **Specialization**: a generic object can be described in more detail using other, more specialized objects of the same type (special cases or sub-types). For example, the New York travel agency is a specialization (a special case) of the Agency-type described in most of the general diagrams, and a large travel agency is a specialization (a sub-type) of the Agency-type described in most of the general diagrams.

Org-Unit



Environment of an org-unit

An *org-unit* generally corresponds to an element defining the organization of the enterprise.

An org-unit:

- can be located at one or more sites.
- sends and receives messages.
- uses one or more applications.
- can be split into sub-org-units and have sub-types.

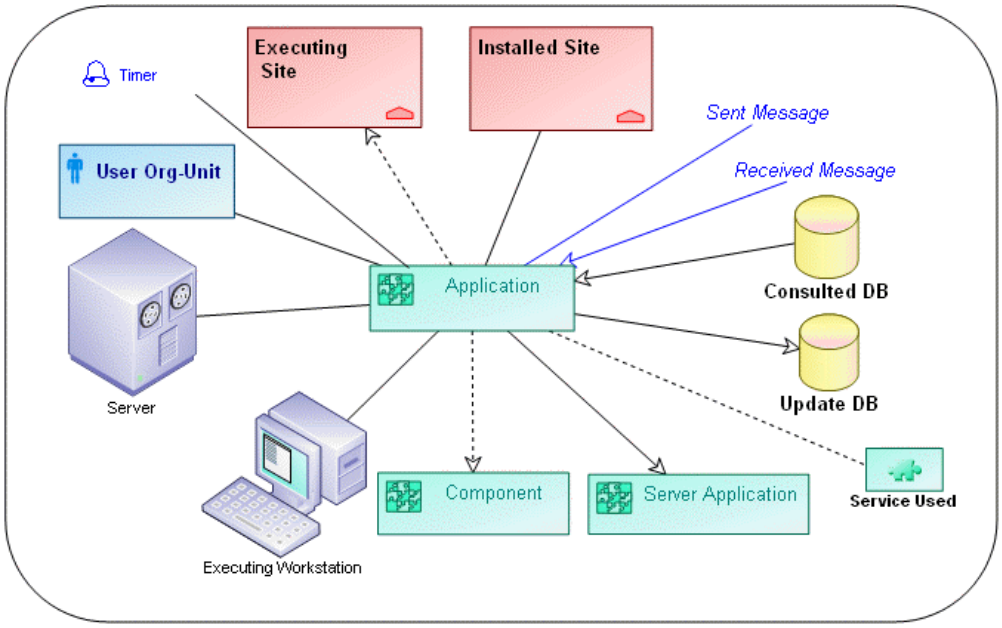
Examples

- Salesperson, assistant, supervisor (workstations).
- Accounting, invoicing (departments).

The org-unit is accessible in the "Org-Units" view of the AAD and TID.

> ☛ *An org-unit can be external. An external org-unit exchanges flows (messages) with the enterprise.*
>
> *Examples: customer, supplier, bank.*

Application



Application Environment

☛ *An application is a set of software tools coherent from a software development viewpoint.*

An *application* sends and receives messages, primarily to and from other applications. These flows are specified in Application Architecture Diagrams.

An application:

- is used by org-units at the sites where it is installed.
- runs on one or more sites, workstations, servers, or nodes.
- can be split into component applications.
- can trigger a server application.
- consults and/or updates a database.
- defines a certain number of services.
- uses services that are part of other applications.

The following can be associated with an application:

- an Application Architecture Diagram
- a Technical Infrastructure
- an Application Tree.

This object can be accessed in the standard views of the AAD, TID and AT.

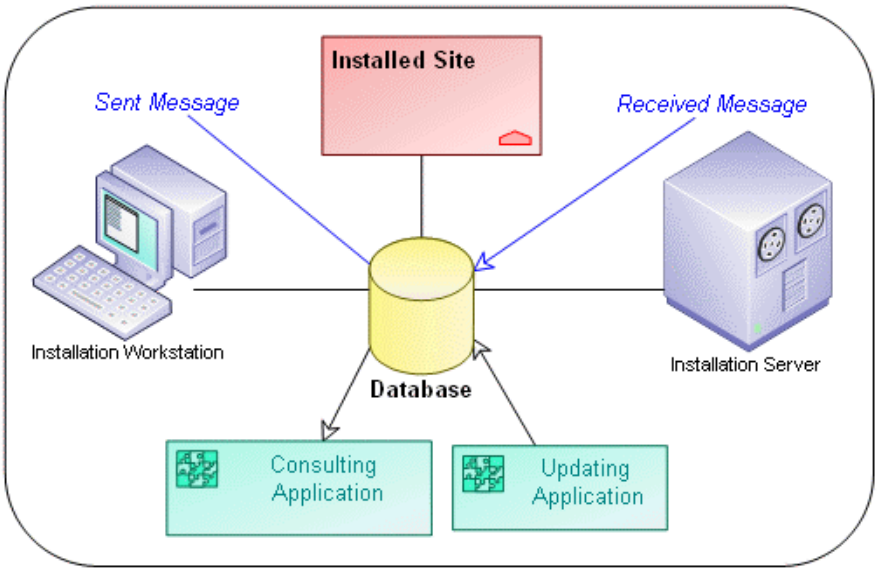
The description of an application can be specified by its type:

- Application type: to specify if the application is a software package, in house application, system software (Windows NT, etc.), middleware software or office system (Word, Excel, etc.).

Examples

- ”Credit” application in the banking industry.
- ”Inventory management” application in distribution.

Database



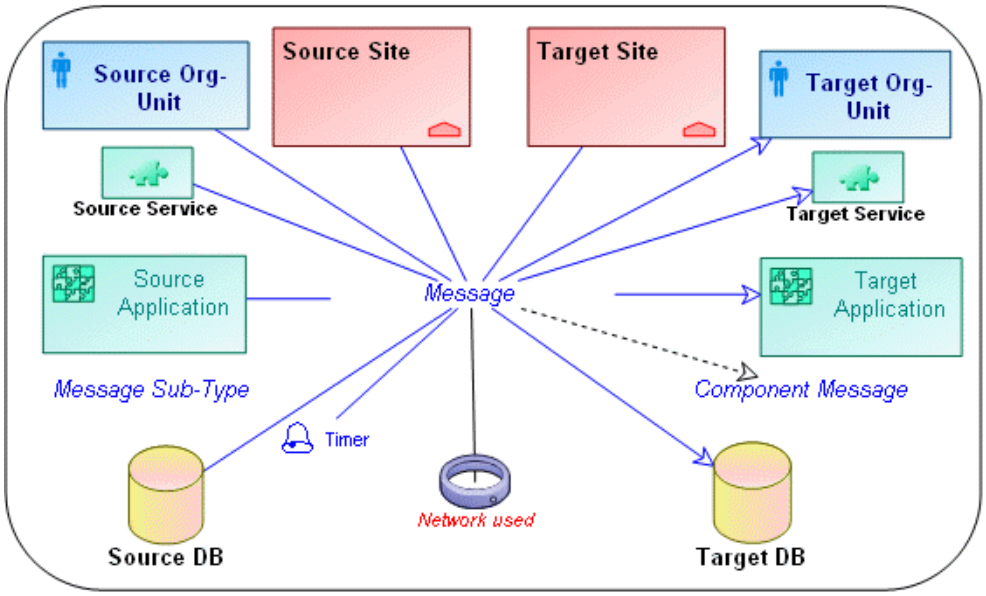
Environment of a database

A *database* is located at a site and installed on a workstation or a server. It is consulted and updated by applications and services. It can be updated by information flows (received message). It allows you to extract data from it (sent message).

This object is accessible in the standard views of the AAD and TID.

☛ *A database enables specification of data logical or physical storage structure.*

Information Flow (Message)



Environment of a message

A message represents information flowing within an enterprise or exchanged between the enterprise and its business environment. Messages can be information flows such as orders or invoices. For convenience, financial and material flows such as payments or product deliveries are also represented by messages.

A *message* can be sent or received by:

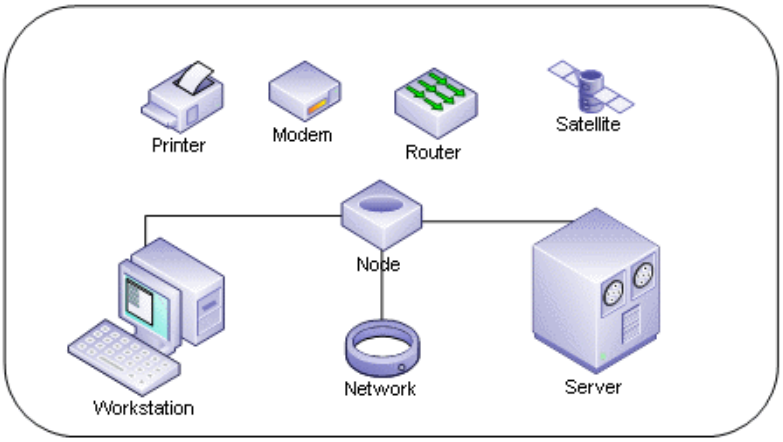
- an org-unit
- a site
- an application
- a service
- a database

A message can be connected to a timer, which specifies a predetermined frequency. You can also indicate the network used by a message.

You can specify that a message is sent only if a predicate is satisfied.

This object is accessible in the standard views of the AAD and TID.

Node



Environment of a network node

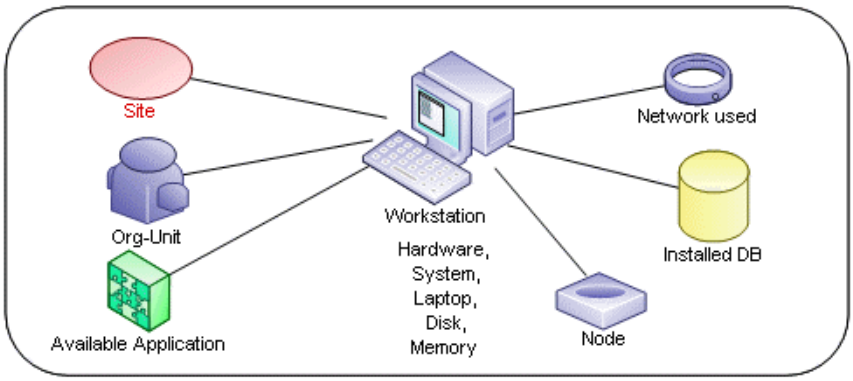
A node is an element of the network such as a printer, modem, or router.

You can connect a *node* to a network, or directly to a workstation or server.

A node can only be accessed in the standard view of a TID.

A node is described by its hardware type: router, printer, modem, etc. You can also add new types of hardware to the predefined node types.

Workstation



Environment of a workstation

Standard workstation that corresponds to a user profile.

In the AAD or TID, when defining workstations, you need to describe the machines that will be assigned to the org-units in your enterprise. Note that you can indicate which applications and databases will be run on a given workstation. This information allows you to make the correct choice in workstation configurations.

You can also indicate the network that connects the workstations and servers,

he nodes of the network such as printers and modems. These nodes are directly connected to the workstation.

This object is accessible in the standard view of the TID.

In an AAD, the workstation is accessible from the technical view of the information system.

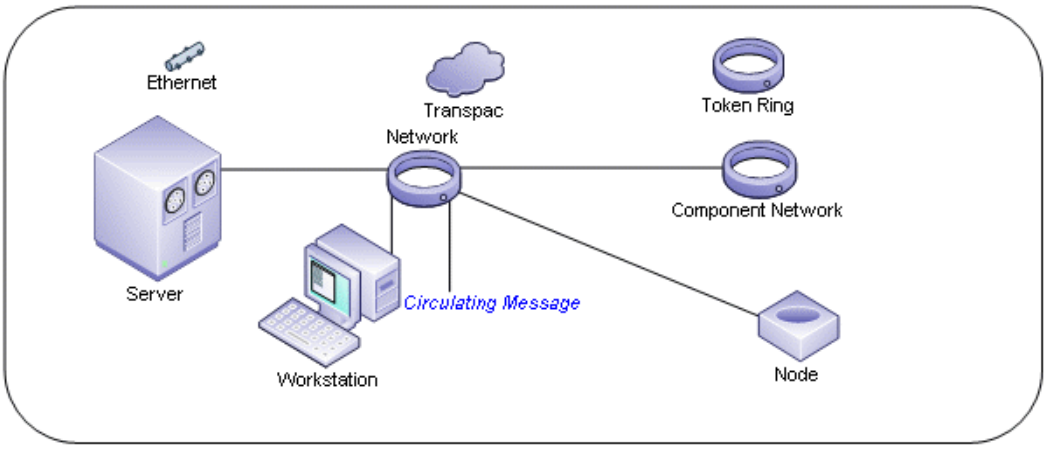
Project

A project is a part of a system whose study is entrusted to the same team.

A *project* can be split into component projects.

The goal of a project can be to model an application.

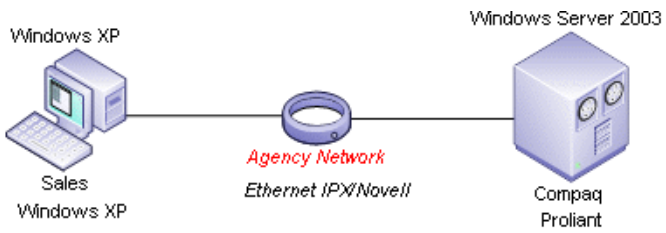
Network



Environment of a network

A network consists of a set of computers that can exchange data. These computers can be geographically remote from each other, remaining connected via telecommunications.

A *network* can also be local, that is, specific to a site in the organization. The network connects servers and workstations in the AAD and TID.



The messages sent between applications or external org-units that travel across the network are also indicated.

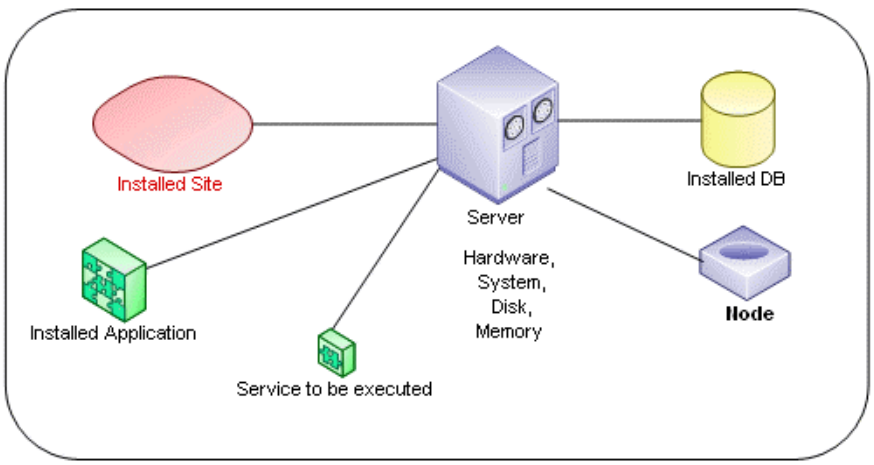
The nodes (modems, printers, etc.) of the network are also described.

You can define the components of a logical network (the protocol) in physical networks.

This object is accessible in the standard view of the TID.

In an AAD, you must use the technical view to access network.

Server



Environment of a server

A computer which provides a service to the users connected to it via a network. This computer can have a database and run Applications.

In the AAD or the TID, you can indicate the *server* on which an application runs, the geographical location of this server at a site and the databases on this server. You can indicate the type of computer to be used, the operating system, required

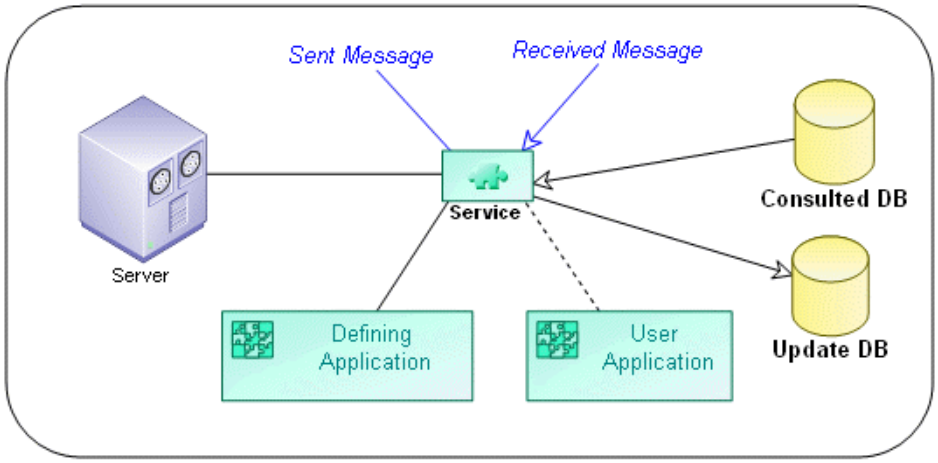
memory and disk capacity, and the network nodes such as modem, routers, printers, etc. that are directly connected to the server.

You can also indicate that a service belonging to an application runs on a server other than the application server.

This object is accessible in the standard view of the TID.

In an AAD you can use the server object in a specific view.

Service



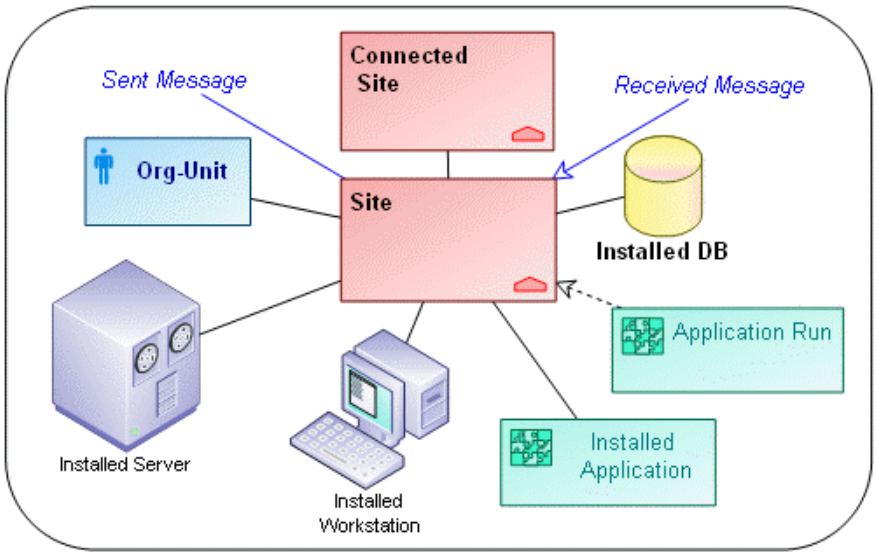
Environment of a service

A service is a component of an application that is available to the end user of the application.

You can specify in each **HOPEX IT Architecture** diagram the *services* that are defined in each application and those that are used by another application.

You can also indicate for these services the databases that they consult or update, the messages that they receive or send and the server on which they run (if they don't run on the same workstation as their application).

Site



Environment of a site

A *site* is an important geographical location for an enterprise because:

- part of the enterprise activity takes place at this location, or
- information such computing data, archives and other media are stored here.

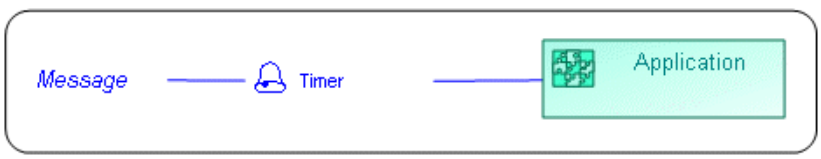
A site can be split into component sites and have sub-types. The organization of a site is determined by its org-units. It can receive and send messages. A site can be connected to other sites. An application can be stored at one site and be executed at another.

The following can be associated with a site:

- A Technical Infrastructure

This object is accessible in the standard views of the AAD and TID.

Timer



Environment of a timer

A *timer* allows you to indicate how often:

- a batch application is triggered, or
- a message is sent.

Examples:

```
Weekly
Twice a month.
Every 100 invoices.
First working day of month.
```

Virtualization and Clustering Modeling

This paper proposes an approach to model virtualization and clustering based on the infrastructure metamodel supplied from the 2009 SP2 release of the Mega modeling tool.

Introduction

What is Virtualization?

Virtualization refers to the abstraction of computer resources. This technique allows the virtually creation of servers, the installation of software packages on these virtual servers and the running of the installed software packages as if they were located on an actual machine. The technique was primarily used to minimize space allocation necessary to store servers, to reduce electricity power the servers consume and so to save money. Other perspectives are the ability to balance from one configuration to another depending on the current load in a completely transparent manner for the end-user.

Need for Virtualization Modeling

The virtualization has progressed so that complete technical environments can be virtually created (servers, networks, sets of application…). On the other hand it becomes more and more difficult to understand where applications and servers are actually located and then to evaluate risks for the business if any technical item fails.

This is the reason why there are more and more requests for virtualization modeling and the ability to deduce impact of virtualized item to the business.

What is Clustering?

A computer cluster is a group of linked computers, working together closely so that in many respects they form a single computer. The components of a cluster are commonly, but not always, connected to each other through fast local area networks and share common storage means. Clusters are usually deployed to improve performance and/or availability over that of a single computer, while typically being much more cost-effective than single computers of comparable speed or availability.

Since the group of clustered computers is viewed as a single one by the end-user the clustering technique can be considered similar to the virtualization technique. But the slight difference here is all the machines may exist … or be virtual. In this context, the performance improvement is due to the grouped computation capabilities while the virtualization objective is to transport end-user machines to the most appropriate location in real-time.

Need for Clustering Modeling

Clustering leads to the same questions as virtualization does: What are the impacts of a server failure on my business? Is this actual server really useful for my tasks? And so on. To answer such questions, the modeling of clustering is also necessary because the business features is attached to the apparent server while the supporting applications are hosted by the clustered servers.

Prerequisites

The purpose of this article is to explain what can be some practices to model both virtualization and clustering. They are two examples of technical infrastructures that can be supported thanks to the resource architecture metamodel introduced in the 2009 SP2 release. The two examples will be developed based on this metamodel and so the sole prerequisite for such implementation is the use of the MEGA 2009 SP2 release. However, the use of a MEGA 2009 SP3 release is recommended because a larger number of items has been included in the MEGA Architecture add-on and among these items the "Virtual Server" artifact that will be used later in this paper.

In order to benefit from this new metamodel, the option "Business Process and Architecture Modeling / Infrastructure Modeling post-2009 SP2" must be activated (Refer to the "Tool / Option" menu command).

For a better understanding of this article, the reader must have some knowledge regarding architecture modeling and the practice of the MEGA Modeling tool. More information can be retrieved on the topics described below in the MEGA Architecture user guide.

The Resource Architecture Metamodel

This section explores the main concepts of the resource architecture metamodel introduced in the MEGA 2009 SP2. The following table lists some concepts of these metamodel. For more explanations, refer to the MEGA Architecture guide.

Concept	Comment
Resource Architecture	A resource architecture is a combination of physical assets and organization configured to provide a capability.
Artifact	An artifact is any element in the physical domain that is not an application or an organizational element (where organizational includes people). An Artifact can represent a physical system, sub-system, platform, component or simply a physical item that has specific attributes.
Application	An application is a set of software tools as seen from the software development viewpoint.
Org-Unit	An org-unit represents a person or a group of persons that intervenes in the enterprise business processes or information system. An org-unit can be internal or external to the enterprise. - An internal org-unit is an organizational element of the enterprise structure such as a department, a service, or a workstation. An internal org-unit is defined based on how detailed you require your view of the enterprise to be (cf org-unit-type). Example: financial management, sales management, marketing department, account manager. - An external entity is an organization that exchanges flows with the enterprise. Example: Customer, Supplier, Government Office.

These four concepts help to model resource architectures that describe solutions designed to support a given capability. An architecture design explains how components are grouped together and communicate each other. The designed solutions include both the technical aspect (equipment and software) and the human resources. In this context, the architecture is composed of items

exposed above and is more focused on the infrastructure part so only the artifact/application part is detailed.

In order to enable the description of interaction in the context of a solution (resource architecture) each inserted item is represented by an intermediate concept referencing it. The figure hereafter shows these intermediate items. This mechanism separates the interactions that equipments or applications perform in more than one context (refer to the "*Artifact Type and Instance*" article for more details on this topic).

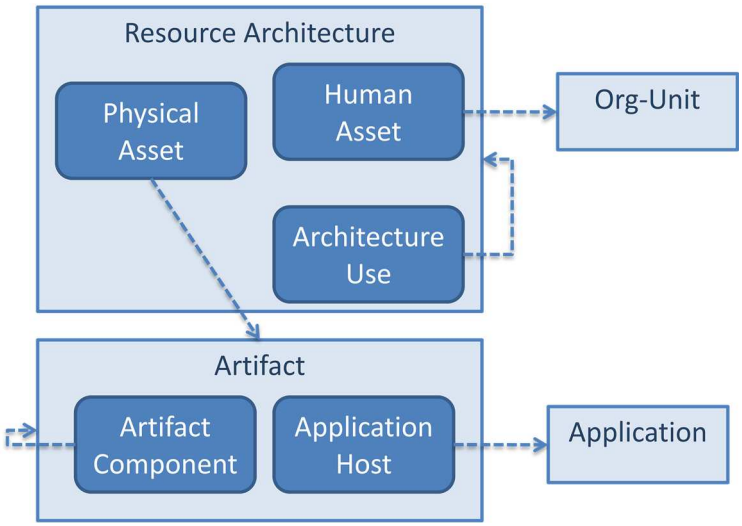


Figure 1: The Concepts Introduced for the Modeling of Technical Architecture Items.

Modeling Virtualization

A first attempt to enable virtualization modeling has been initiated in the MEGA 2007 release. The objective was to enable the identification of virtual servers and the application "generating" such servers. So, an association has been added that links applications to servers (instance of the metaclass Server). This link can be drawn in a technical infrastructure diagram and so belong to the technical infrastructure metamodel introduced prior to the 2009 SP2 release.

The link was sufficient to express facts such as: "the VMWare application creates this virtual server". Servers linked to an application can be identified as virtual servers since they are generated by a virtual machine application. Since the virtual server is a server regarding the type, it can be used as any other servers to host applications.

This principle has not been reintroduced in the metamodel proposed from the 2009 SP2 release for several reasons:

- The simple link between applications and servers is not enough to model all virtualization configurations. Depending on the context, virtualization can be handled by an application, an operating system, a dedicated server and even a cluster of servers. In these cases, numerous associations should be necessary and even an intermediate object to group the potential virtual server generators.
- The new infrastructure metamodel is not defined at the same level. Servers are not described as instances of a dedicated metaclass but as artifacts inheriting from a reference

Server artifact. This new design principle induces that a link between artifacts is available for any artifact even if virtualization has no meaning for some of them.

So, the question now is how to model virtualization based on the Artifact metamodel? Two facts are relevant in such modeling:

- This server is virtual
- This virtual server is generated by this technology (operating system, dedicated servers…).

Identifying Virtual Servers

To model the first topic, one way consists of starting from an existing artifact. The MEGA Architecture add-on contains some artifacts that can be used as starting points. Among them there is the *Server* artifact. From the 2009 SP3 release, the add-on contains also a *Virtual Server* artifact that inherits from the *Server* artifact. Every new virtual server is then a new artifact inheriting from the *Virtual Server* artifact. The variation mechanism can be used to created a family of virtual servers and so refine their definitions.

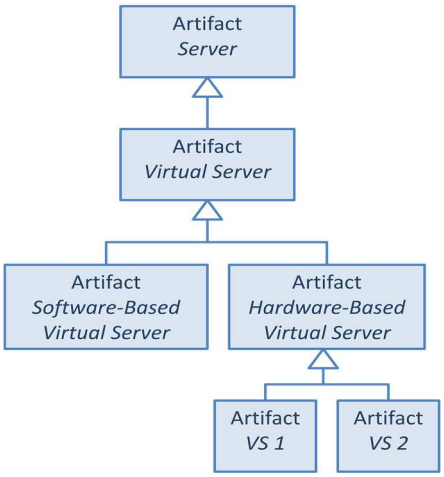


Figure 2: Example of Virtual Server Taxonomy.

This first step allows the identification of the virtual server. Once created, they can be retrieved using a query on variable objects (or more precisely on artifacts). Such a query, named "*Variants of an Object*", is supplied from the MEGA 2009 SP3 release. The query looks like:

```
Select [Variable Object] Where [Variant Side Variation].[Varied Object] Deeply = &"Variant"
```

Processing the query will ask the user for a variable object. Then the *Virtual Server* artifact is selected and the query lists all the virtual servers in the repository. If a more specialized virtual server is selected, the query lists the corresponding subset of virtual servers.

The same identification principle can be applied to any artifact. Indeed, the virtualization technique is available for servers but also for more complex architectures. For example, a virtual network can be created or a complete virtual architecture including a network and a set of servers. For these two examples, a *Virtual Network* artifact is designed as a starting point for all virtual servers while a *Virtual Architecture* instance of the Resource Architecture concept matches the starting point of all

virtual architecture items. Then, variants of the virtual architecture are described with their own components.

Using Properties to Model Virtual Servers

Another way to identify virtual servers is simply by the addition of a customer property on the *Server* artifact. For example, the property type can be named *Virtual* and the available values for this property can be *yes* or *no*.

The advantage of this simple proposal is that the type can be reused for different artifacts (server, network …) or resource architecture. The research of virtual servers is deduced from the previous query but adding the property value condition. Such a query looks like:

```
Select [Artifact] Where ([Variant Side Variation].[Varied Object] Deeply Like "Server") And
        ([owned property].(([Property Value]="yes") And
         ([Property Type].ShortName Like "Virtual")))
```

There are at least two drawbacks with this proposal:

- The query to search for virtual item is more complicated and the ERQL request written above does not manage all the possible cases. Indeed, since the property values can be inherited from varied ancestors, the value is not always connected directly to the artifact. In consequence the query must be updated to take this inheritance into account. This is not the purpose of this article to expose how to go through inherited properties but the fact is some code must be used instead of the ERQL syntax to perform a complete search (vb script, javascript or java code, refer to the MEGA API).
- The second point is that the proposal does not allow the definition of taxonomy of virtual servers has the variation does and then to simply ask for servers, virtual servers, hardware virtual servers... Instead, each sub-family should be implemented thanks to additional properties making the search tool again more complex.

For the reasons exposed here, the use of a dedicated artifact is recommended compared with a solution based on properties.

Including Virtual Servers in an Architecture

Whatever the way the virtual servers are created, they result to some artifacts that can be used in any solution architecture or embedding artifacts describing what are the environments letting the virtual servers available. The common example is some virtual servers hosted by a physical server. An approach can be described by the following steps:

1. Create the virtual servers as previously explained adding the applications exposed by these virtual servers.
2. Create the physical server from the *Server* artifact.
3. Insert the virtual servers in the physical one as Artifact Components.
4. Establish the links between the virtual servers to explain how they interact - if they do so - in the context of the physical server. Such links can be communication channels with communication protocols.
5. Detail the applications that the physical server hosts. Among these applications there is probably some virtualization tools (e.g. VMWare). At this point, if the virtualizing tool is

considered as a platform technology it is more appropriate to model it based on an artifact as are operating systems. One can create the corresponding application later and insert it in the platform technology as an *Application Host*.

Figure 3: An Example of Some Virtual Servers.

In the example above, two servers are available for the end user: *VS 1* and *VS 2*. They are both virtual servers hosting business application (e.g. Invoice). The infrastructure supporting the end-user servers is composed of a physical server *Server XYZ* that generates the virtual server and hosts an instance of the *VMWare* application. The implementation of such design in the MEGA 2009 SP3 Release is based on the *Artifact Assembly Diagrams*. These diagrams are very simple from a graphical representation perspective and look like the following:

Figure 4: The Artifact Assembly Diagram of the *Server XYZ*.

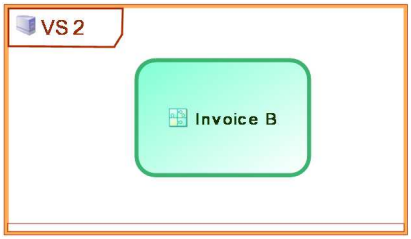


Figure 5: The Artifact Assembly Diagram of the Server *VS 2*.

The principle exposed here can be replicated to any virtual structure. A complex virtual architecture composed of a network, *n* servers and some hosted applications is described the same way: the physical server contains all the virtual items including the network, the virtual servers and the applications that are all artifacts or applications.

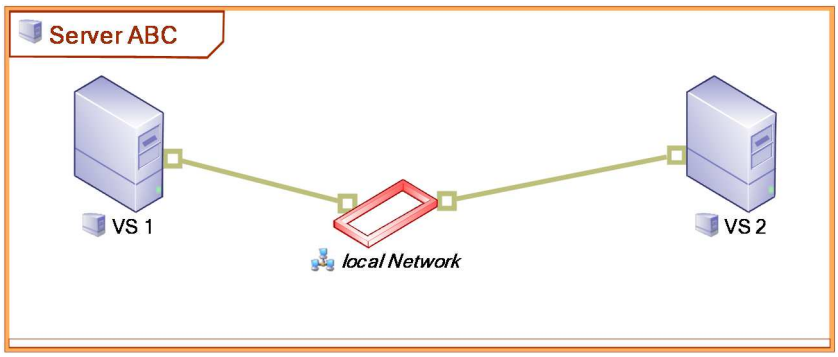


Figure 6: An Example of a Complex Virtual Configuration Hosted by the Server *ABC*.

Modeling Clustering

The modeling of clustering is based on the same principle proposed for the virtualization. A cluster is a group of servers that share common resources to run an expected application faster than each individual server is able to do. For the end-user, there is only one server and only one application he wants to use.

In this context, the modeling approach is the following:

1. Create all the physical servers that are grouped into a cluster. Each server inherits from the *Server* artifact.
2. Create the cluster as an artifact inheriting from the *Server* artifact.
3. Include all the primary physical servers in the cluster as Artifact Components.
4. Add the applications hosted by the cluster that are available for the end user.
5. Add the potential applications of each individual server that are useful for the clustering.
6. Create the links between the physical servers that explain what the communication means are and what is exchanged in the context of the cluster.

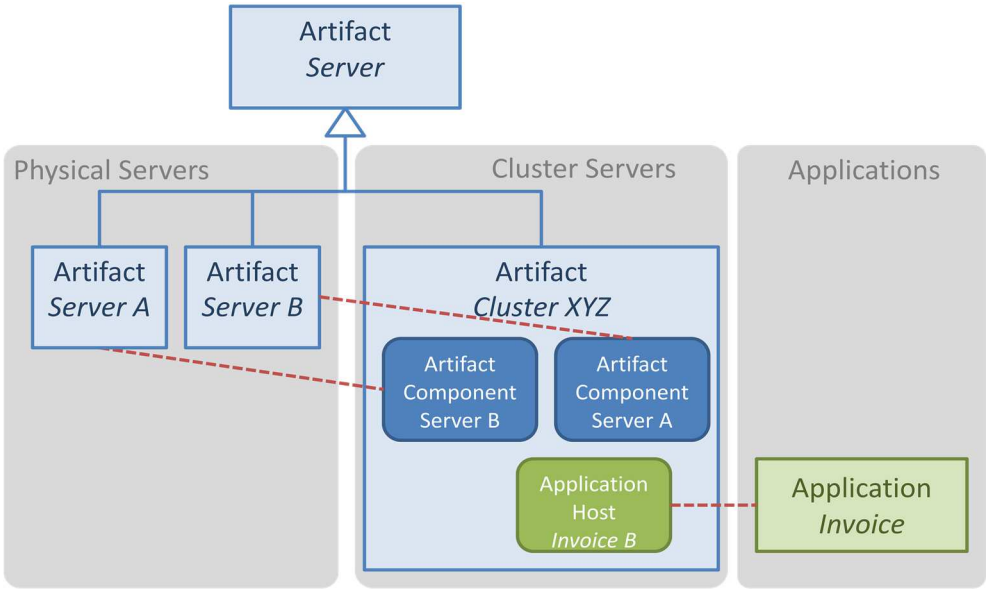


Figure 7: Example of Clustering Model.

In the example above, the cluster server *XYZ* hosts the application *Invoice* from the end-user perspective. Depending on the current load, the application is balanced either to *Server A* or *Server B* but this is totally transparent for the end-user. At least the model explains the architecture and then it is possible to deduce that the application *Invoice B* can be impacted by a failure of the physical server if the architecture describes some communications channel between the application and the servers.

Mixing Virtualization and Clustering

Finally, it is possible to have configuration including virtual items and clusters. For example, one can create a cluster of virtual servers … or some virtual cluster of servers! Based on the proposed approach there is no problem to design both these combinations. Virtual servers are simply grouped in a cluster server and in a physical server. The reusability of the artifacts is powerful in this context and it is not mandatory that the virtual and cluster group contains the same items.

Figure 8: A Cluster of Virtual Servers.