Dr. Marcel Schlatter
IBM Distinguished Engineer
Member of the IBM Academy of Technology

University of Zurich IBM

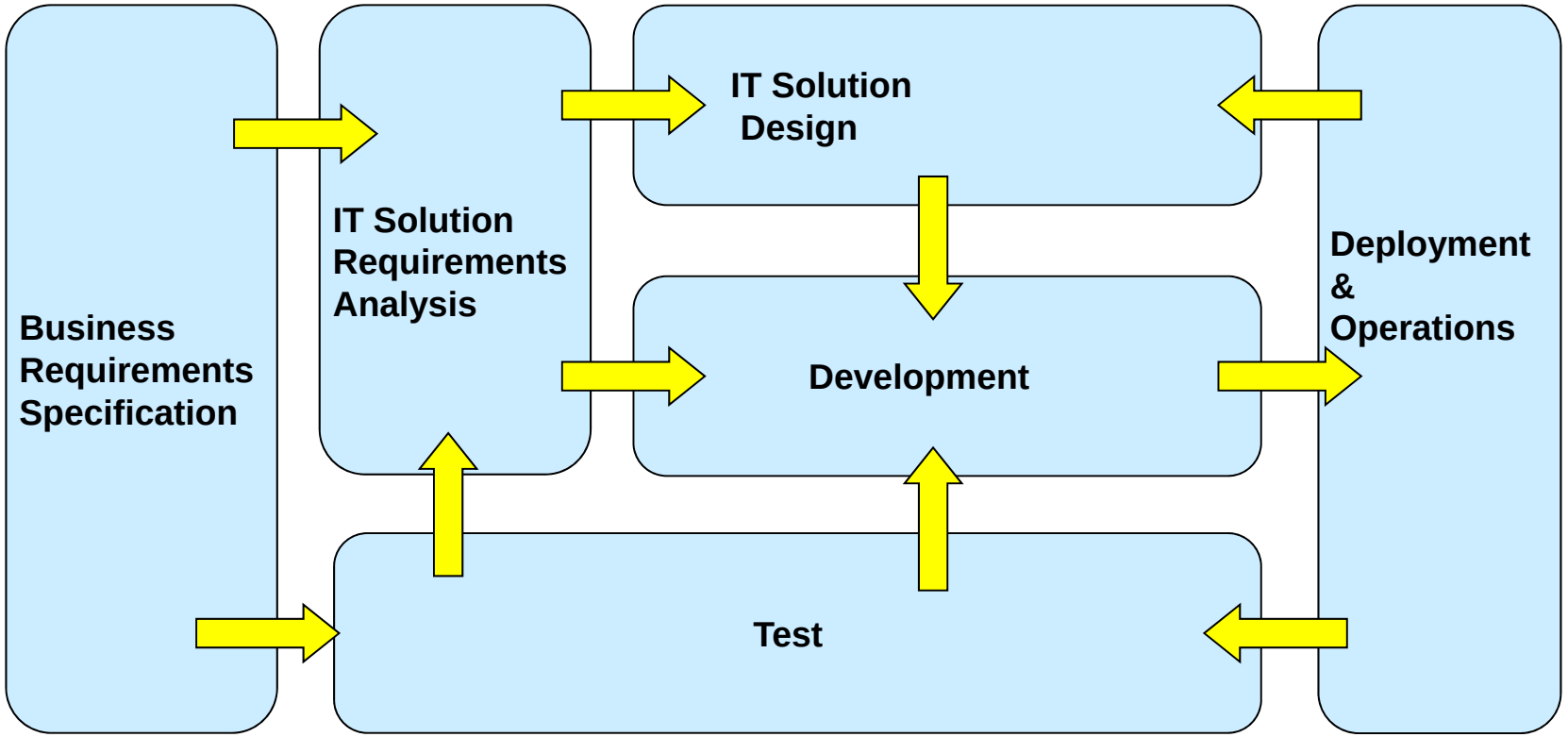
Developing a Solution's IT Architecture

Work product creation using a "top down", requirements driven approach

Separating concerns: organizing the requirements and design into distinct parts

Incrementally developing business requirements and their IT solution

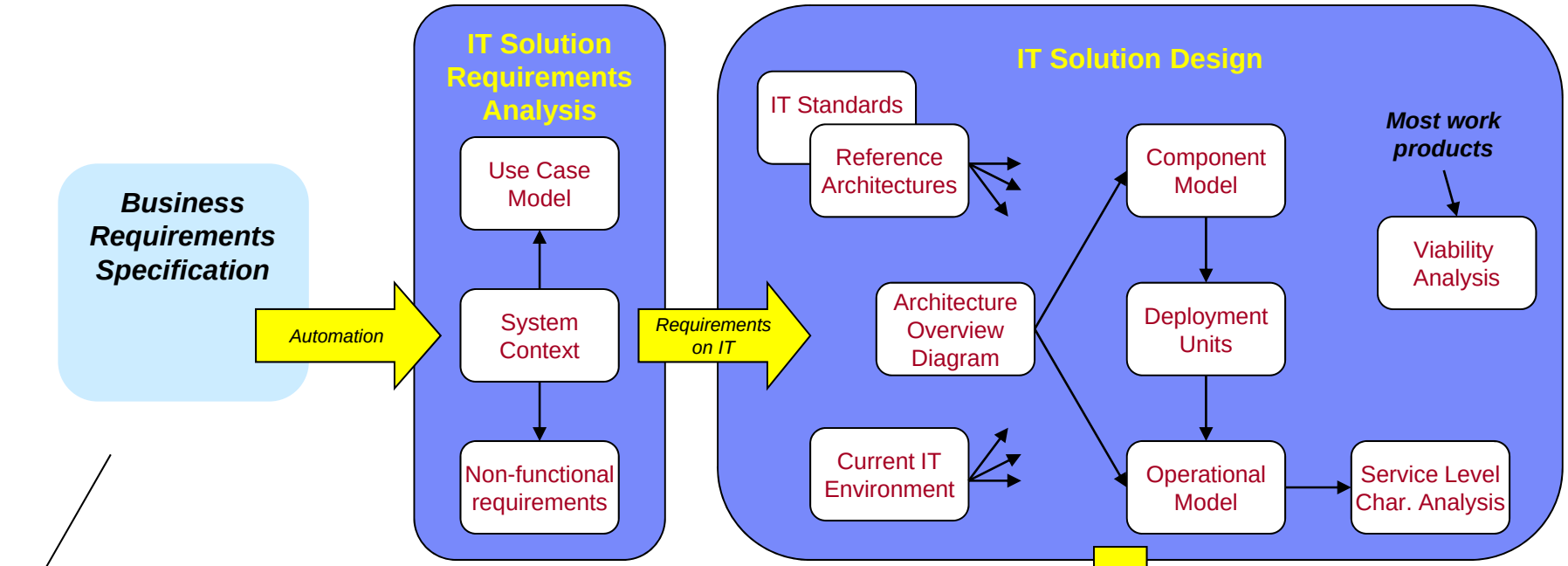
© IBM Corporation 2011

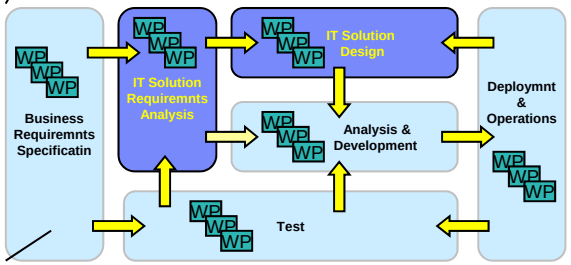


© IBM Corporation 2011

Defining and documenting the various aspects of the IT solution's requirements and design is achieved by using a set of **IT Architecture work products**, each focused on a specific view of the IT system

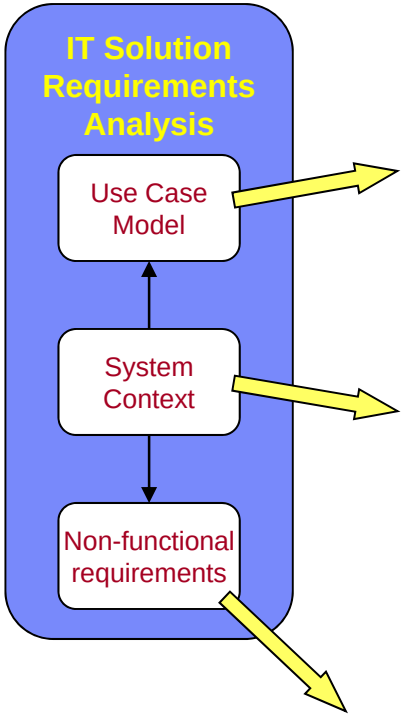
Separation of concerns

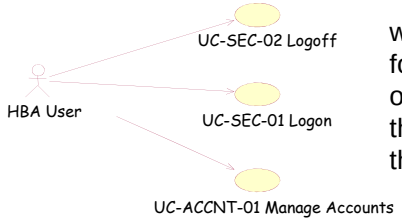




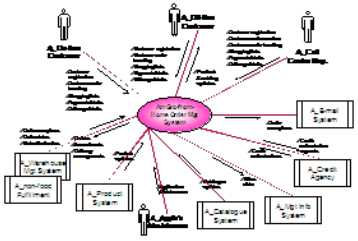
© IBM Corporation 2011

The IT architect uses three core work products to document the business requirements their IT System will support…





"Use Case Model" defines what the IT system must do for all its business (and other) users – it describes the system behaviour at the "automation boundary"

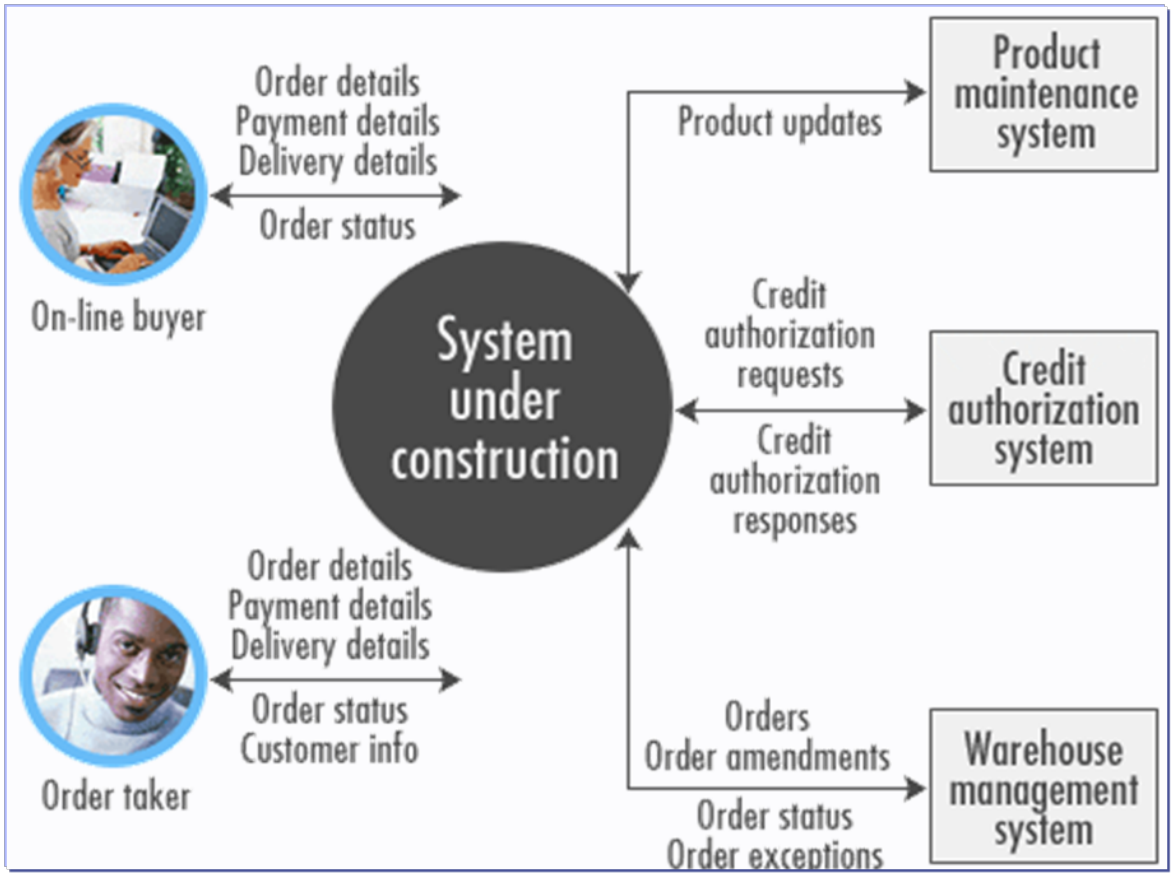


"System Context" defines the boundaries of the to be designed system, documenting the external users and other IT systems with which this system must interact

"Non functional requirements" describe the characteristics of the IT systems functionality, such as transaction response times and workload volumes

© IBM Corporation 2011

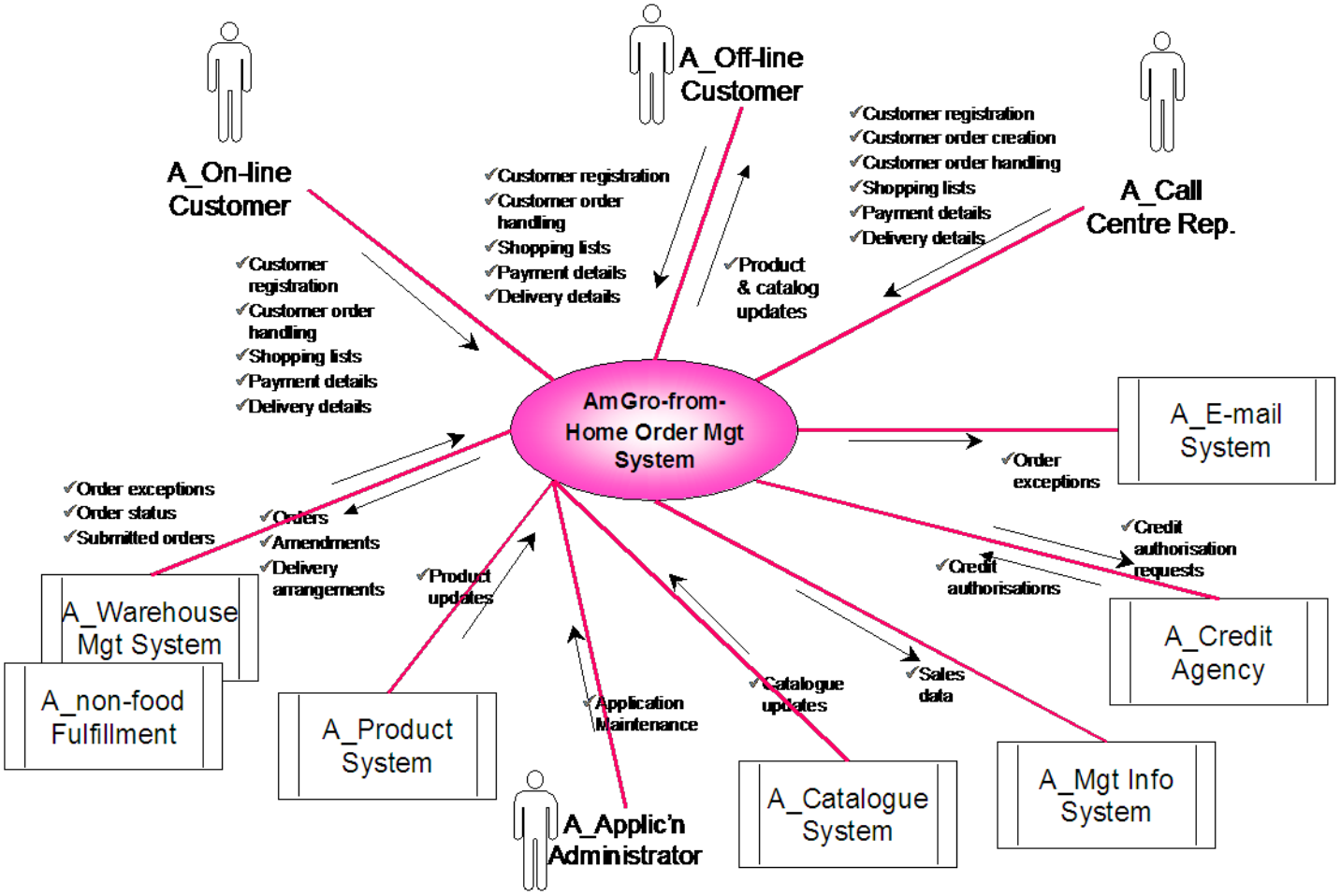
The System Context is essential to capturing the scope of the project



- **The System Context helps to:**
  - Clarify the environment in which the system has to operate
  - Put bounds on the system
  - Identify external interfaces (users or systems)

© IBM Corporation 2011

System Context

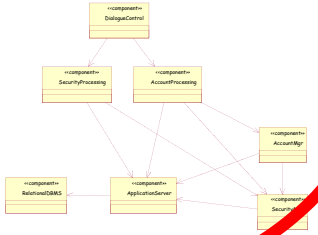


© IBM Corporation 2011

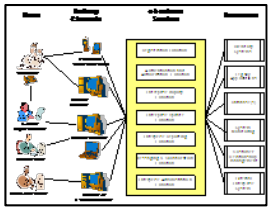
The IT architect uses four core work products to document and communicate their IT system's design

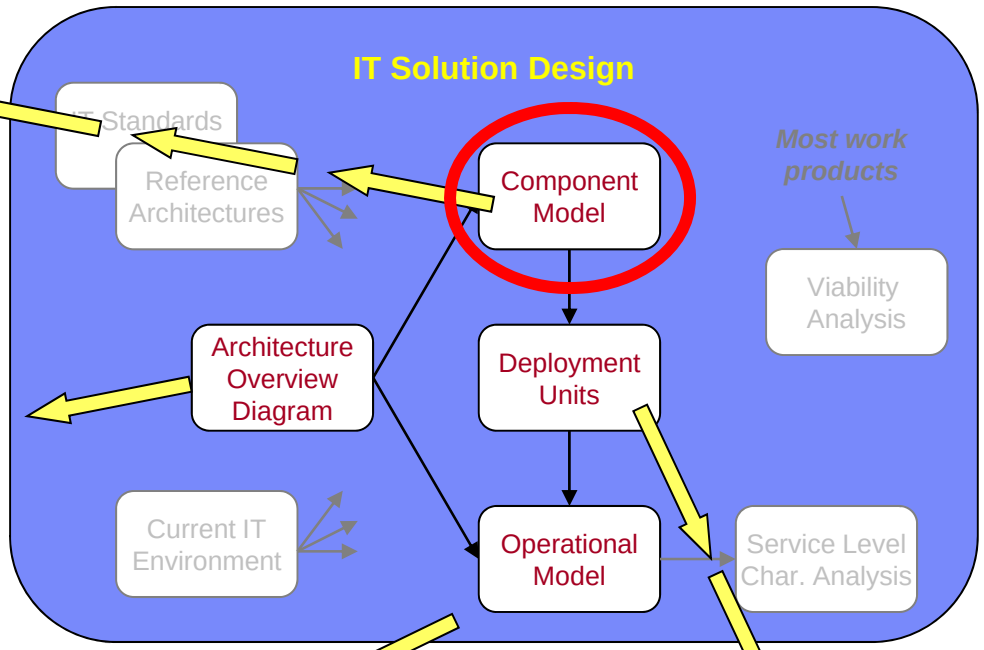
Application Architect - Software Engineering

"Component Model" describes the structure of an IT System in terms of its software components with their responsibilities, interfaces and relationships, and the way they collaborate to deliver the required functionality.

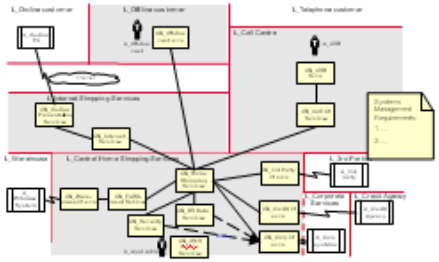


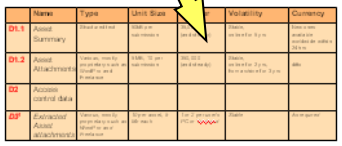
"Architecture Overview Diagram" provides a picture (not a model) of the whole IT system "on a page" as a means of communicating the salient points of the design. AODs are audience specific





"Operational Model" defines the organisation of the IT system across locations, documenting the placement of the solution's components onto nodes connected across the organisation, in order to achieve the solution's operational NFRs

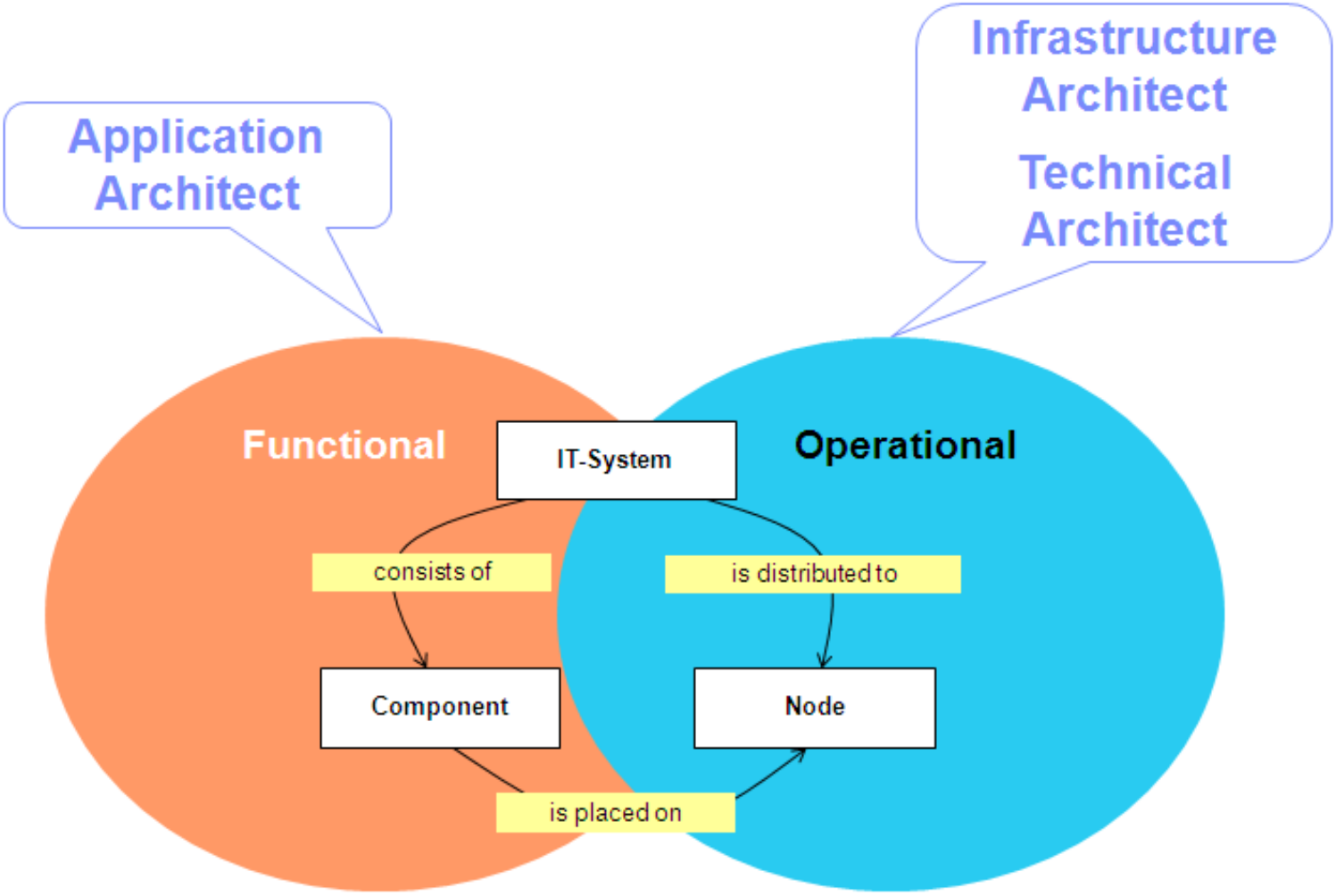




"Deployment Units" represent various aspects of components, as a convenient means of documenting their non functional requirements, as well as their placement across the Operational Model

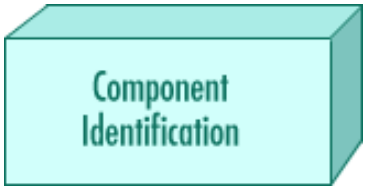
Infrastructure Architect - Systems Engineering

© IBM Corporation 2011

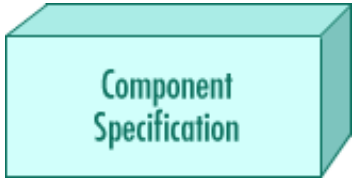


© IBM Corporation 2011

The Component Modeling technique consists of three steps…



- Partition into subsystems and components and assign responsibilities
- Review architectural patterns, reference architectures, and reusable assets
- Structure ensuring loose coupling, high cohesion, and so on

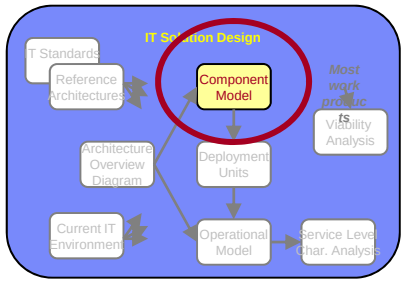


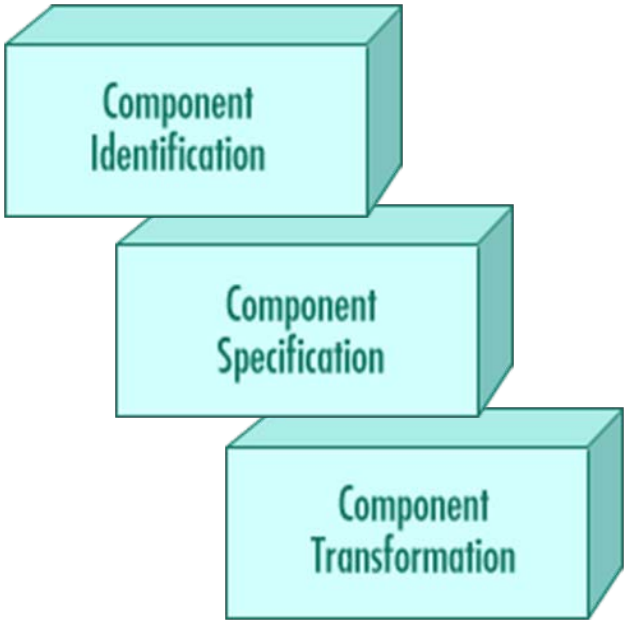
- Specify interfaces
- Specify operations and signatures
- Specify pre- and post-conditions

- Identify products and packages
- Define implementation approach

© IBM Corporation 2011

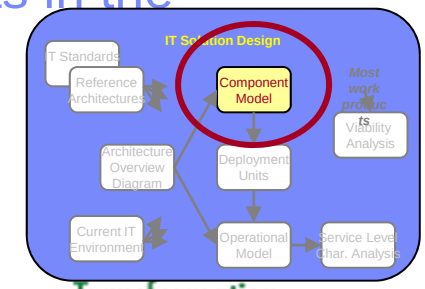
…which are performed iteratively

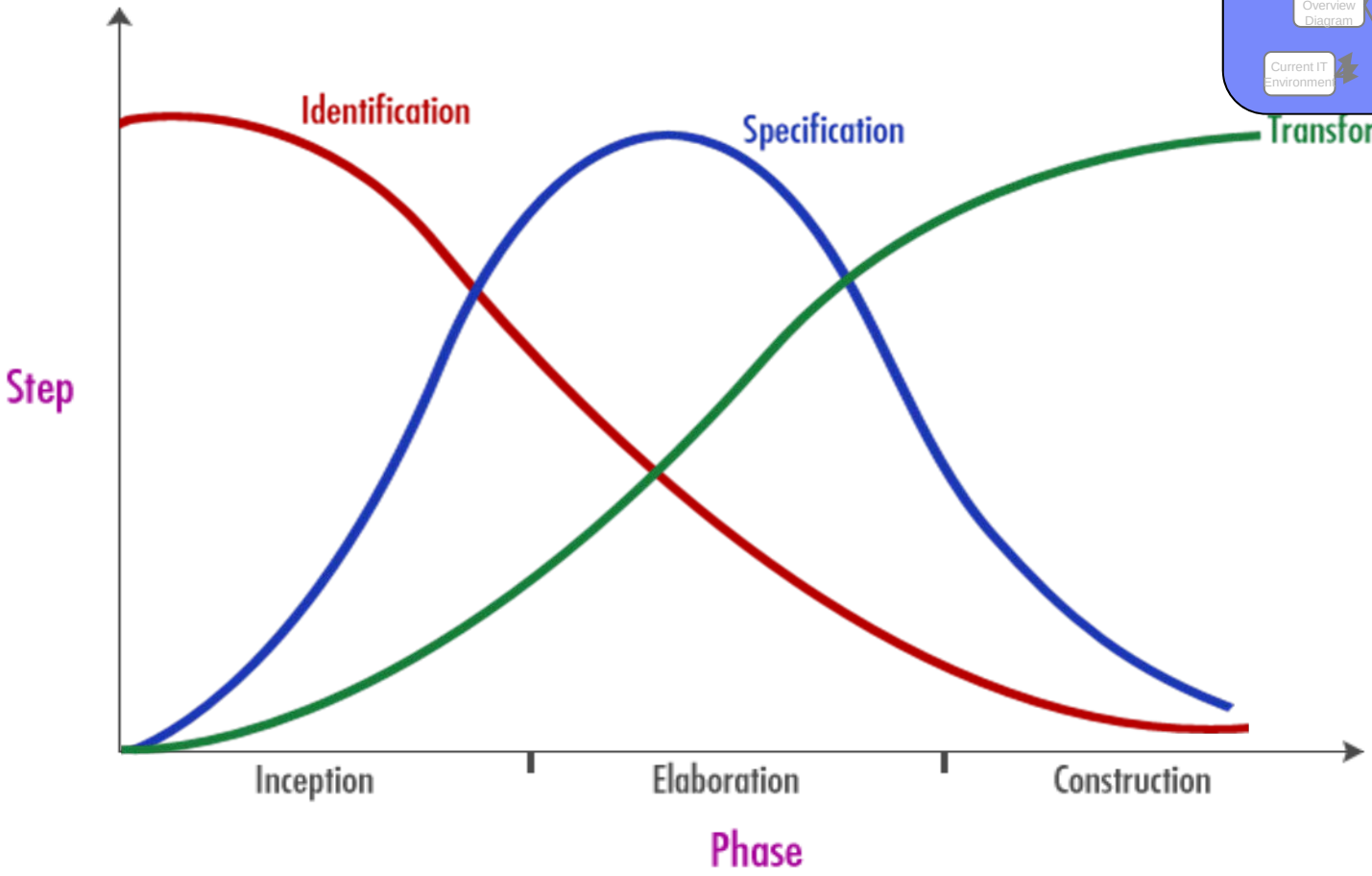




© IBM Corporation 2011

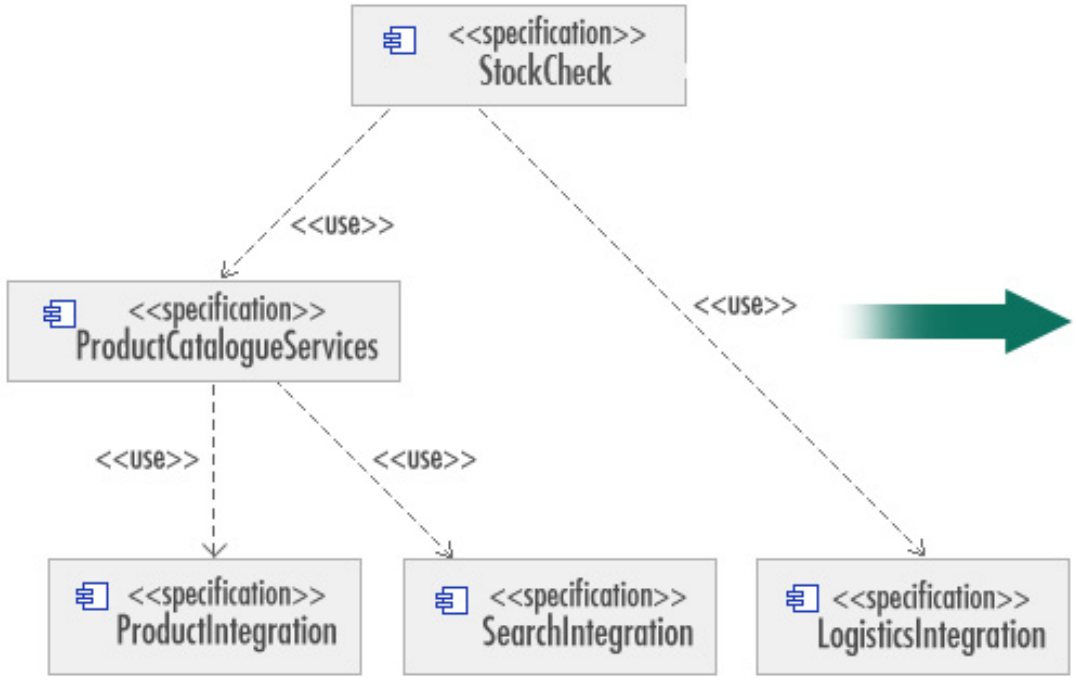
Each step is applied, to varying degrees, at different points in the delivery process





© IBM Corporation 2011

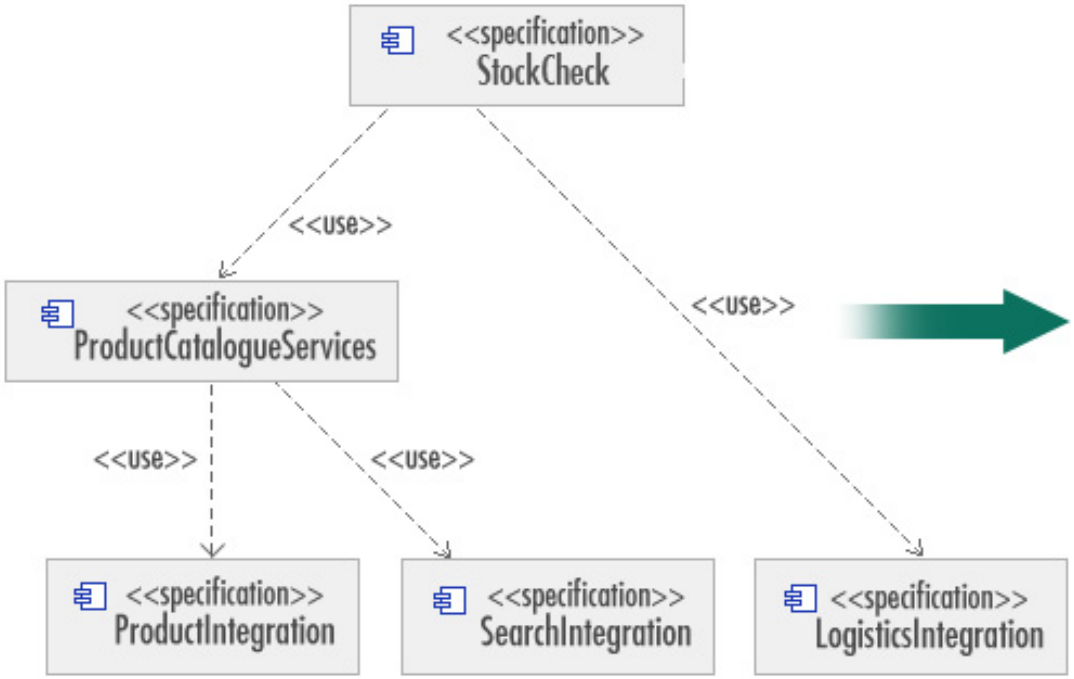
The Component Model



- **Bridge the gap between the requirements (the “what”) and the solution (the “how”)**
- **Visualize and help understand the system**
- **Specify the logical structure or behavior of the system**
- **Document decisions made**
- **Allow placement decisions to be made about where components will execute**

© IBM Corporation 2011

The Component Model is used as input into a number of activities

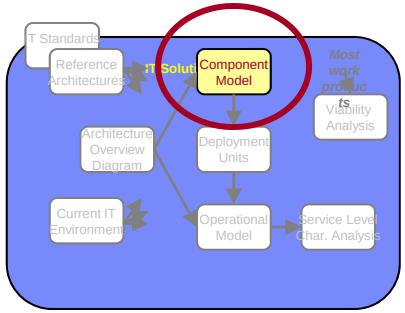


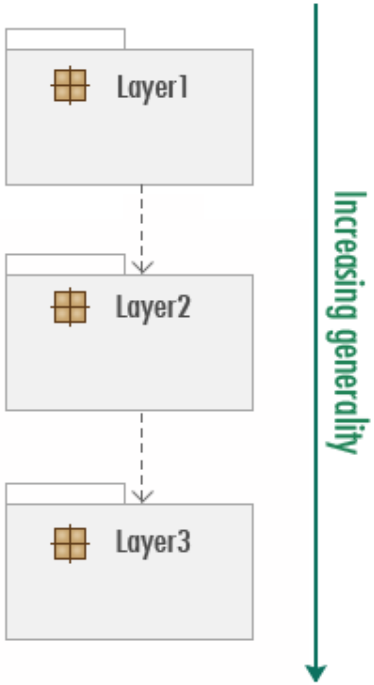
- **Work Allocation**
- **Version Control**
- **Design Strategy**
- **Reuse**
- **Testing**
- **Project Management**
- **Product/Package Selection**

© IBM Corporation 2011

Component Modeling often involves placing components into layers

- **Layering provides a logical partitioning of components into a number of sets (layers)**
- **Rules define relationships between layers**
  - Strict: Components only depend on components in the same layer or the one below
  - Non-Strict: Components may depend on components in any lower layer
- **Layering provides a way to restrict inter-component dependencies**
- **Well-layered systems are more loosely coupled and therefore more easily maintained**

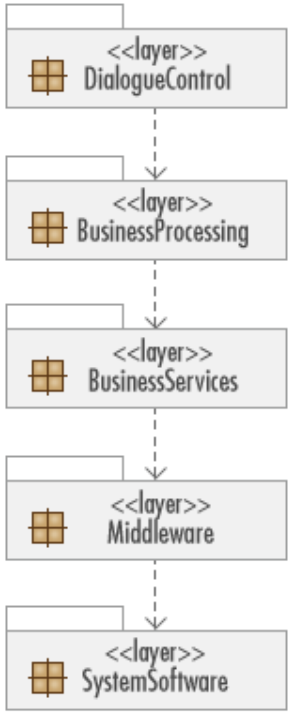




© IBM Corporation 2011

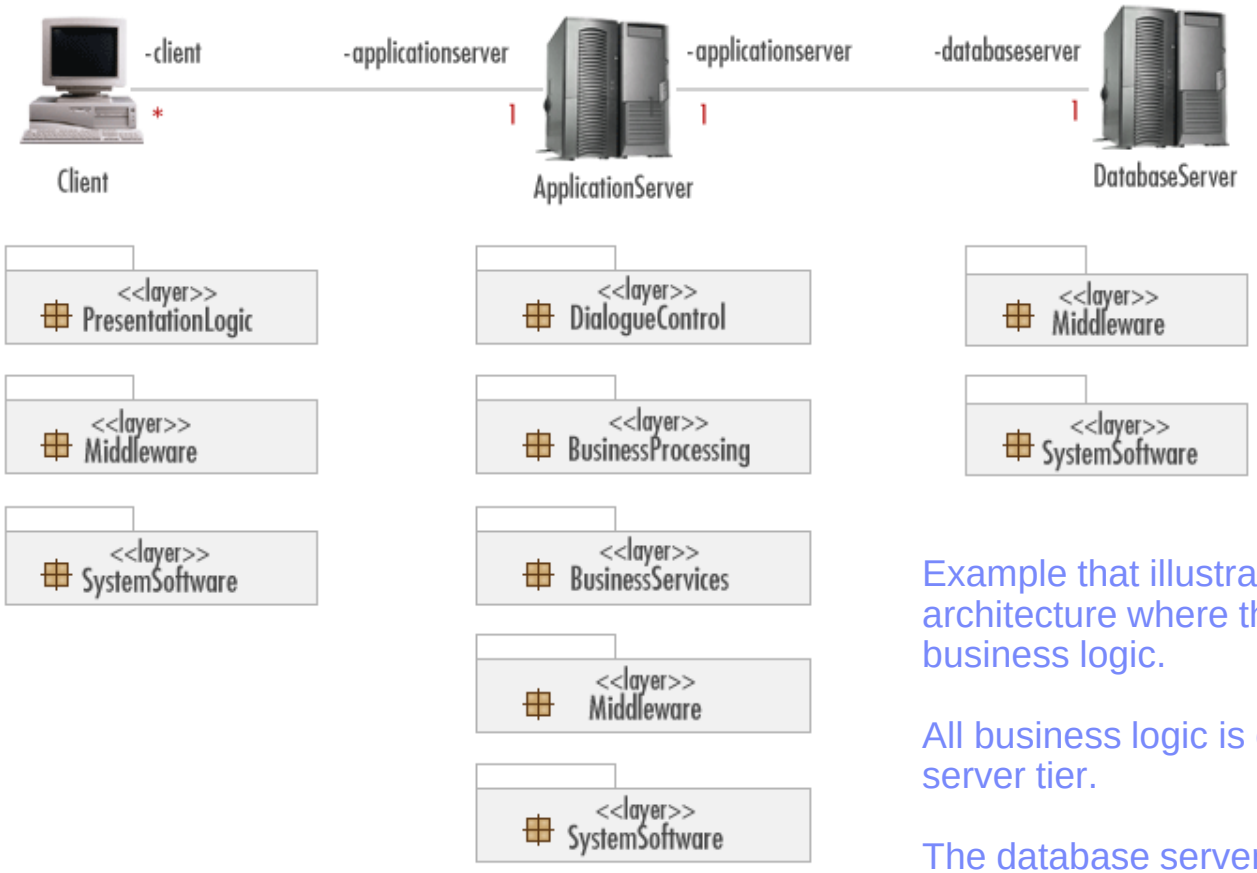
An example of layered architecture

- **The dialogue control layer handles user-system interactions and use case logic**
- **The business processing layer contains application-specific services that handle use case step logic and choreography**
- **The business services layer contains more general business components that may be used in several applications**
- **The middleware layer contains components such as interfaces to databases and platform-independent operating system services**
- **The system software layer contains components such as operating systems and databases**



© IBM Corporation 2011

In a Multi-Tier System, each Tier can be layered independently



Example that illustrates a three-tier, thin-client architecture where the client contains no process or business logic.

All business logic is on the middle, application server tier.

The database server just contains middleware (that is, the database and communication software).

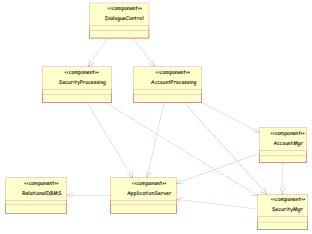
All tiers contain system software (such as an operating system)

© IBM Corporation 2011

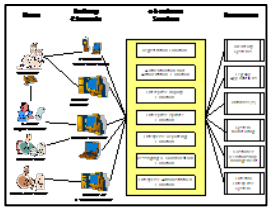
The IT architect uses four core work products to document and communicate their IT system's design

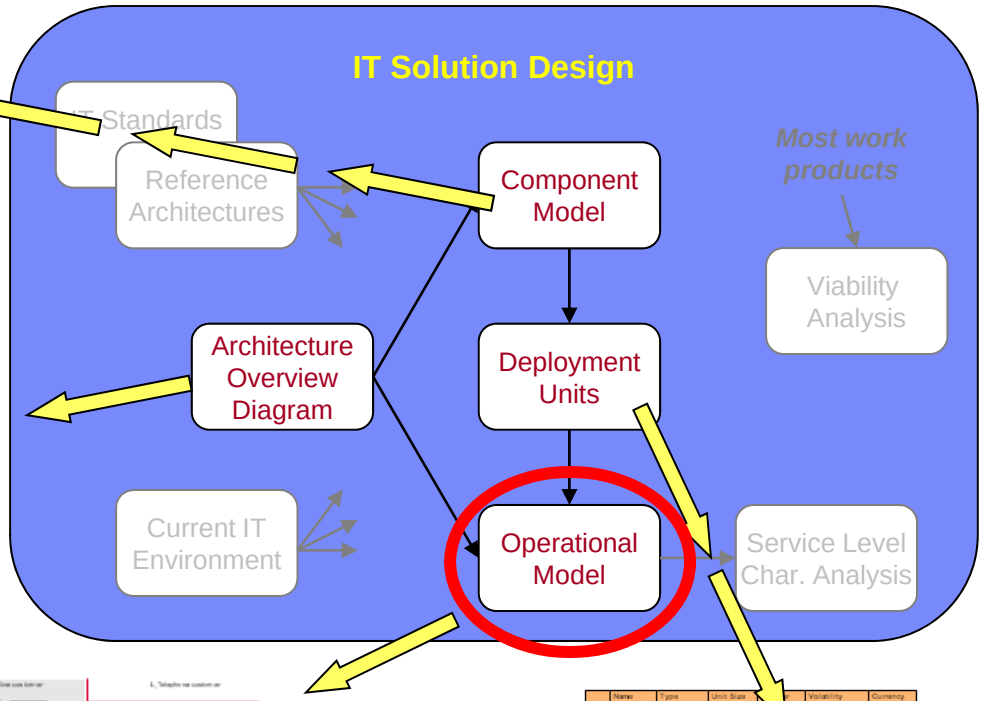
Application Architect - Software Engineering

"Component Model" describes the structure of an IT System in terms of its software components with their responsibilities, interfaces and relationships, and the way they collaborate to deliver the required functionality.

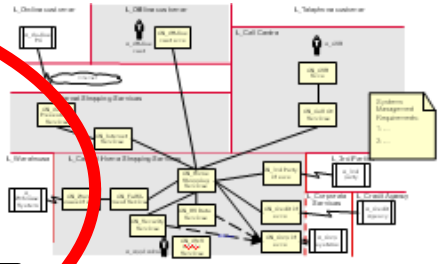


"Architecture Overview Diagram" provides a picture (not a model) of the whole IT system "on a page" as a means of communicating the salient points of the design. AODs are audience specific

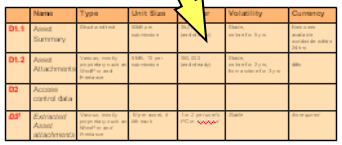




"Operational Model" defines the organisation of the IT system across locations, documenting the placement of the solution's components onto nodes connected across the organisation, in order to achieve the solution's operational NFRs



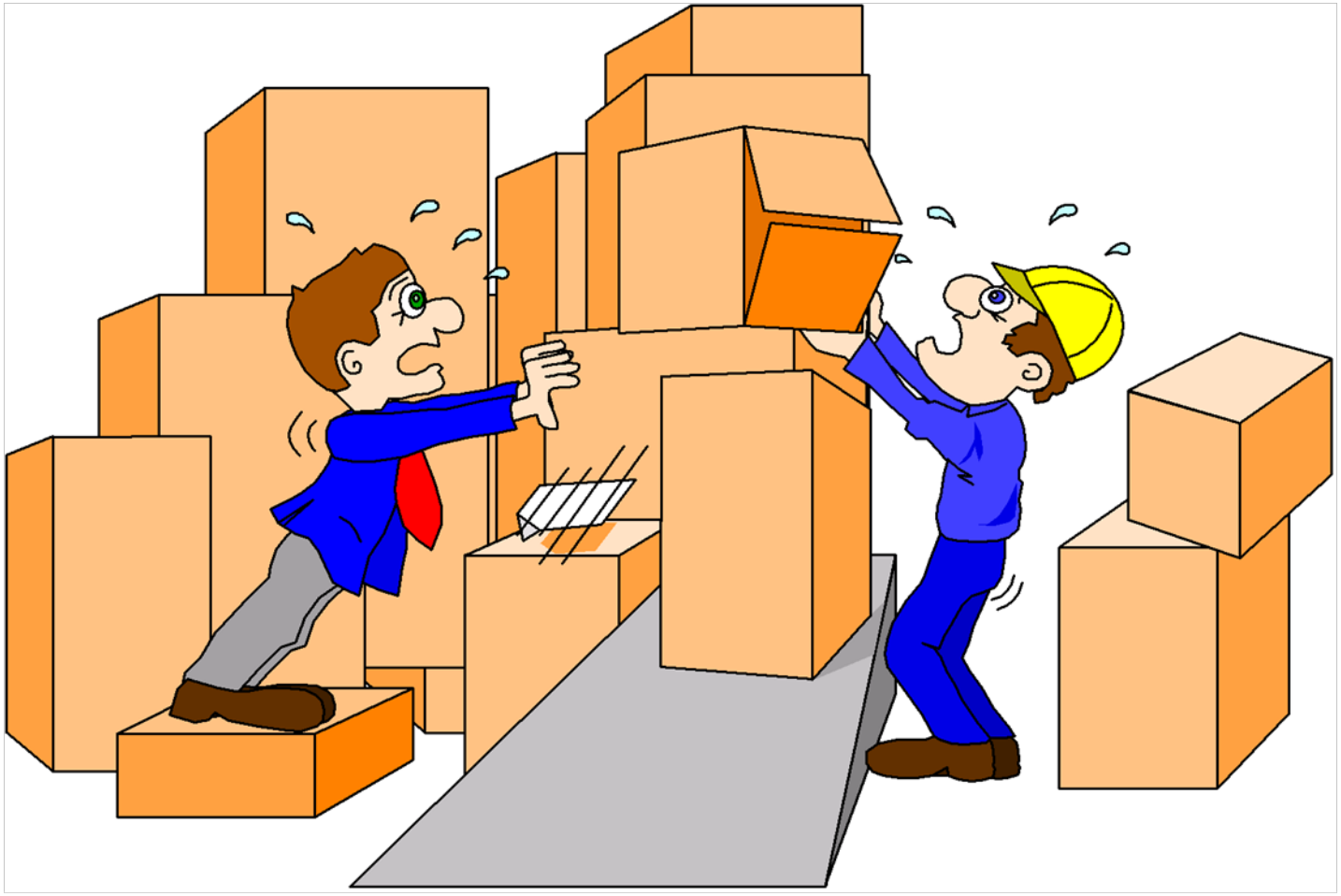
Infrastructure Architect - Systems Engineering



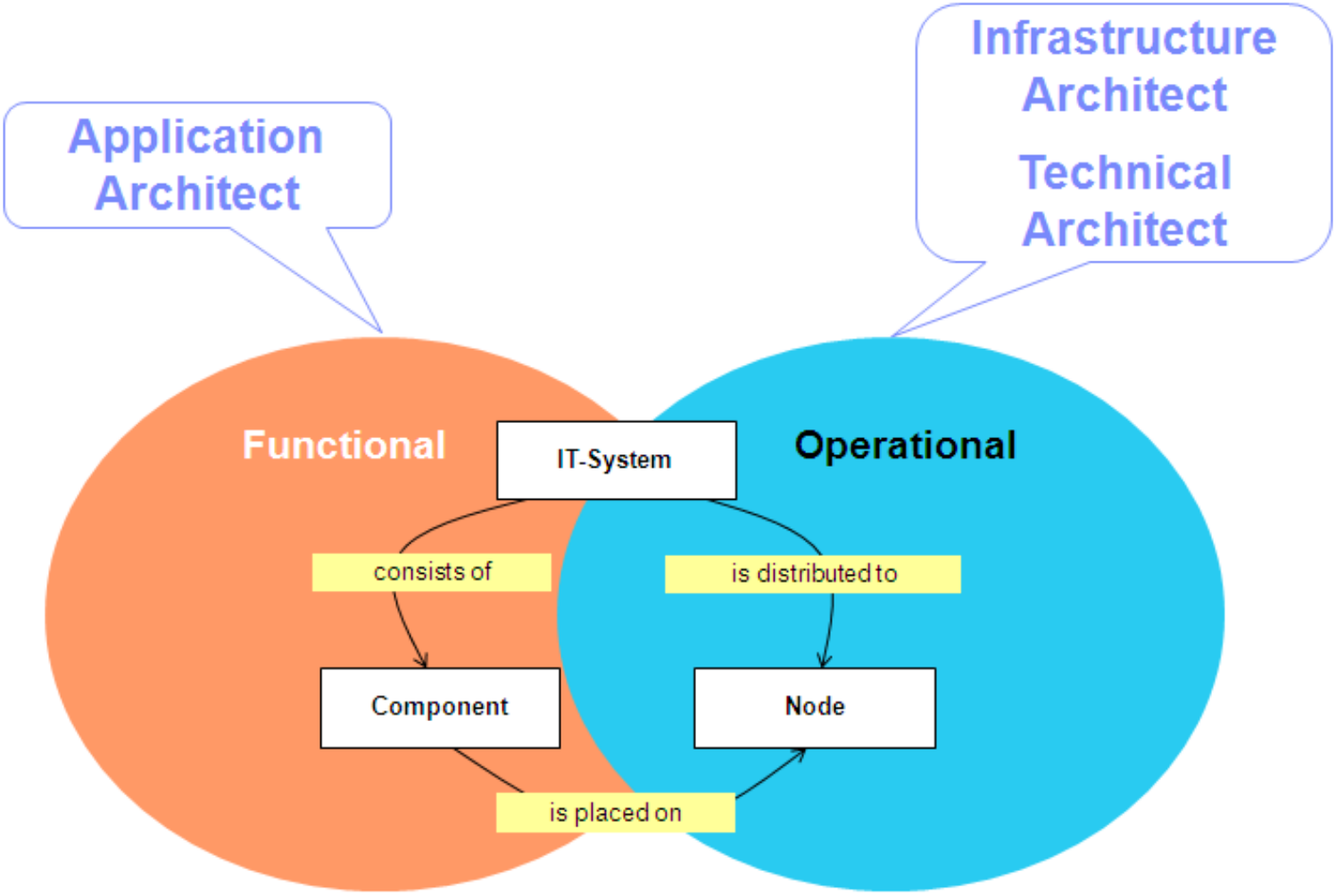
"Deployment Units" represent various aspects of components, as a convenient means of documenting their non functional requirements, as well as their placement across the Operational Model

© IBM Corporation 2011

Interactions between functional and technical considerations



© IBM Corporation 2011



© IBM Corporation 2011

The Operational Model represents the system's "infrastructure architecture", using a variety of model elements

- The geographic structure of the locations and their borders, over which the IT system will be deployed and operated
- The placement of the system's nodes into these locations
- The deployment of the system's components across these nodes, using deployment units
- The connections between nodes
- The organisation of the system's elements into zones
- Sizing and other hardware specifications for all the computers, storage devices and network technologies

© IBM Corporation 2011

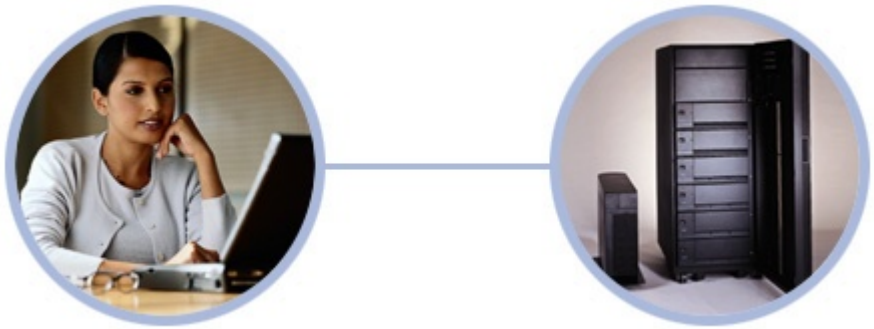
Operational Model
How do we decide where a system's components should go?

Let us consider a simple example: a "single component" system…

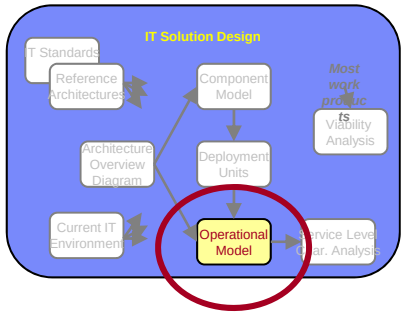
…Microsoft Word

Let us think about what we have to do, when "deploying" Word onto a very simple environment:

- Where does "Word" go?



What is it we have to deploy? Where does "it" go? Let us sketch up some ideas…



© IBM Corporation 2011

So, for our simple WORD example, we should first identify:

(1) That the Word component has the following deployment units:

DU	Description	Characteristics (e.g.)
P1_WYSIWYG_Display	Microsoft Word desktop UI	Minimum screen size: 1024 x 768
E1_Word	Microsoft Word execution	Required operating memory: 512 MB Minimum CPU (equiv): 1Ghz
D1_User_Documents	Word documents being edited by the users	Typical document size: 5 MB Typical active documents: 100 Some documents are critical to business operation

(2) There will be two locations

- **L_Branch_Office**, which represents where Word users work
- **L_Central_site**, which represents a IT services data centre

(3) And two nodes

- **N1_Office_Workstation**, which represents a Microsoft Windows PC
- **N2_Central_Server**, which represents a Microsoft Windows Server

© IBM Corporation 2011

There are many ways of deploying a single component into a simple system…

- **Option 1**

- Presentation
- Execution
- Data
- Installation

Word, running on XP, stand-alone

- **Option 2**


- Presentation
- Execution
- Data
- Installation

- Word, running on XP, with remote file serving

- **Option 3**


- Presentation
- Data
- Execution
- Installation

Word, running on Citrix, with local data

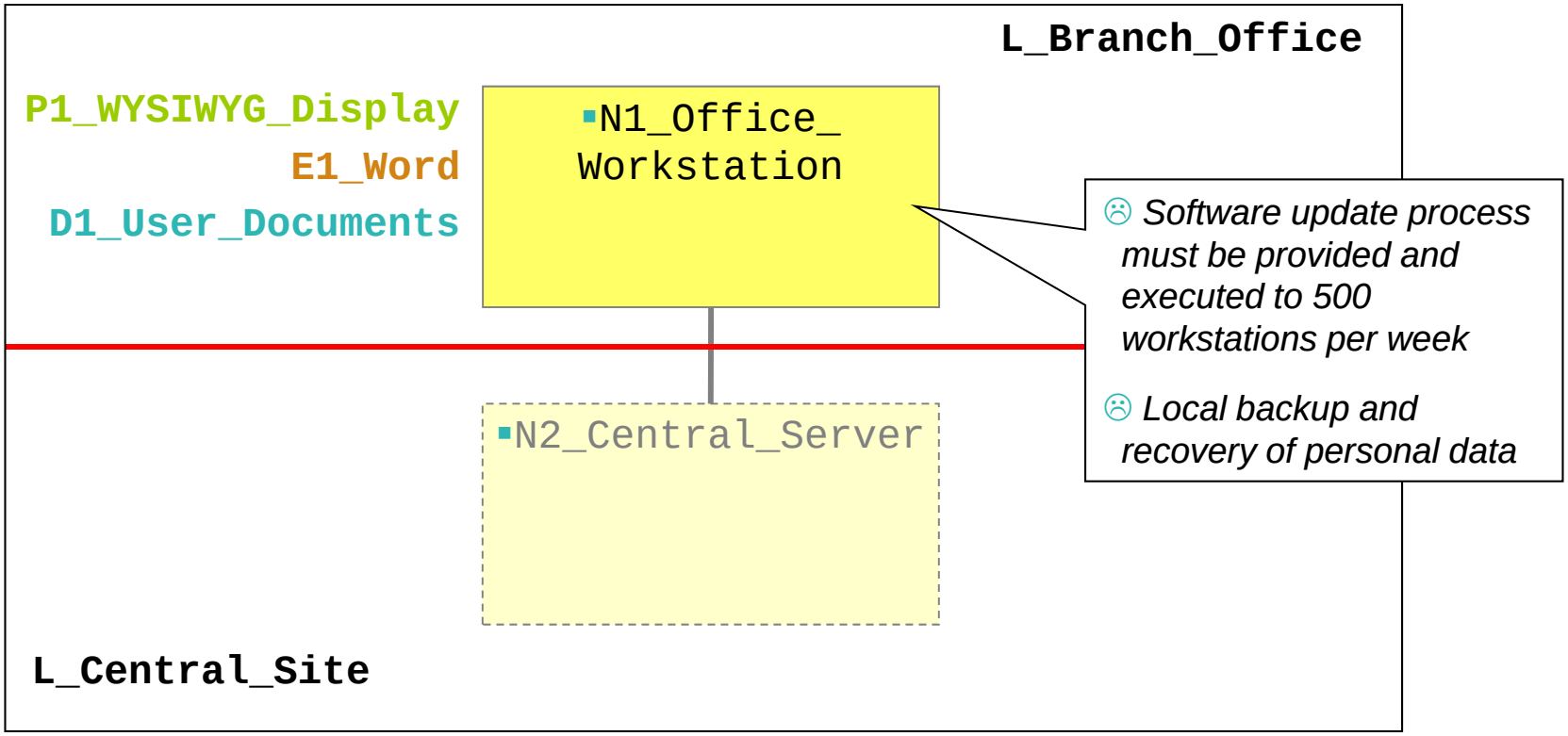
- **Option 4**


- Presentation
- Execution
- Data
- Installation

Word, running on Citrix, with remote file serving

© IBM Corporation 2011

Option 1 - a local installation, with all DUs on the Office workstation…



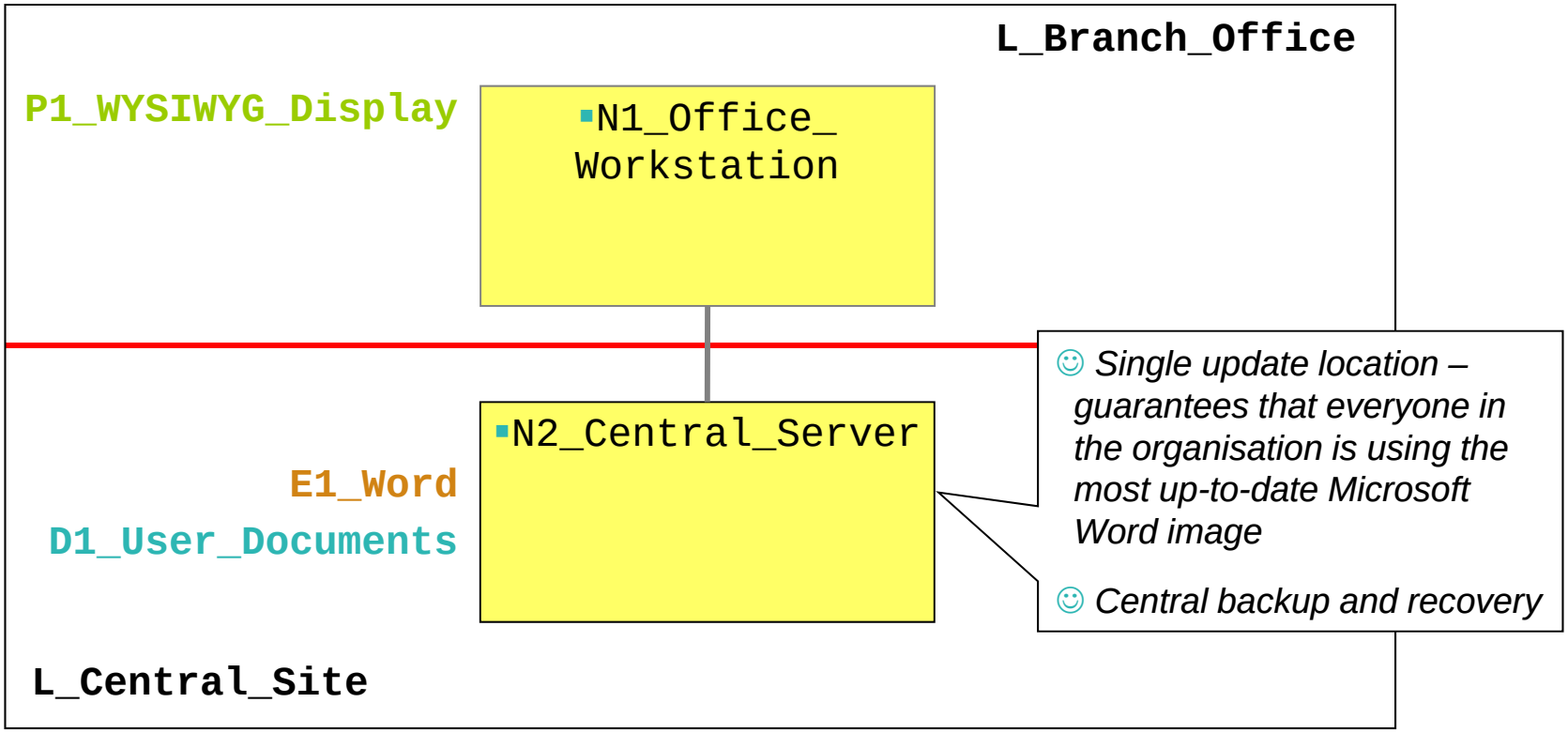
But this approach has many systems management issues:

- Software updates becomes an issue: e.g. regular security patches issued by Software vendor
- Backup and recovery the responsibility of the end user
- Number of users and number of branches

Reconsidering the placement decisions leads to…

© IBM Corporation 2011

Option 4: using a pattern based on a server-side installation with data served remotely



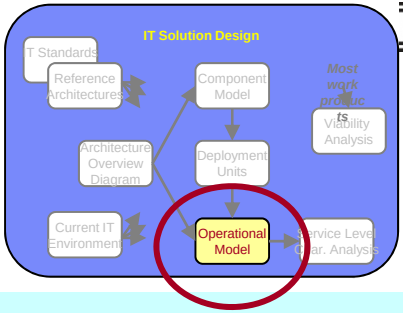
This is a much more manageable software update and data management regime.

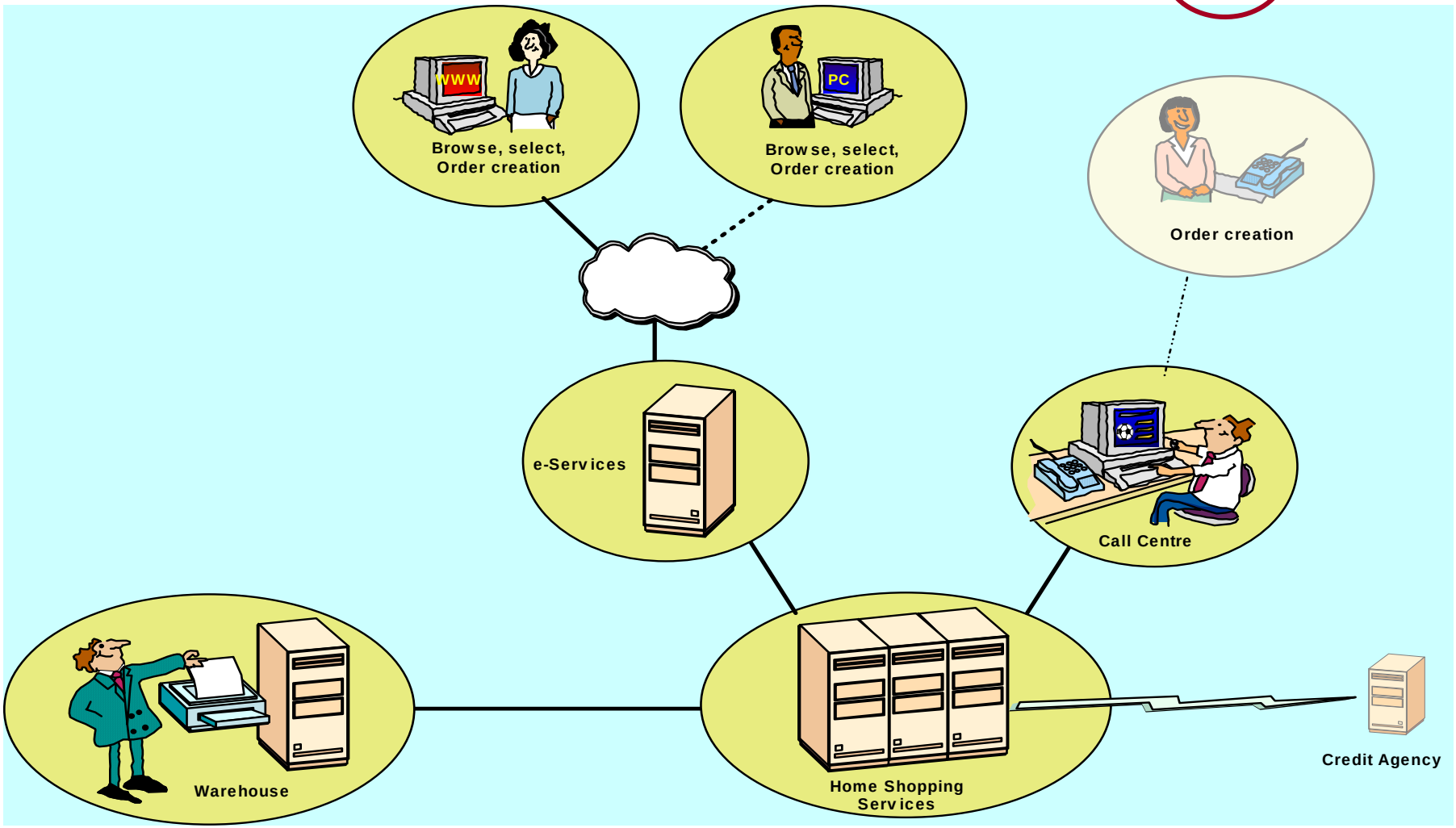
However…

- …desktops may not have appropriate remote file server capability…
- …not all end users may be able to easily access central servers…
- …some workstations may be unable to support remote execution…

© IBM Corporation 2011

Operational Model -- Geographic Background

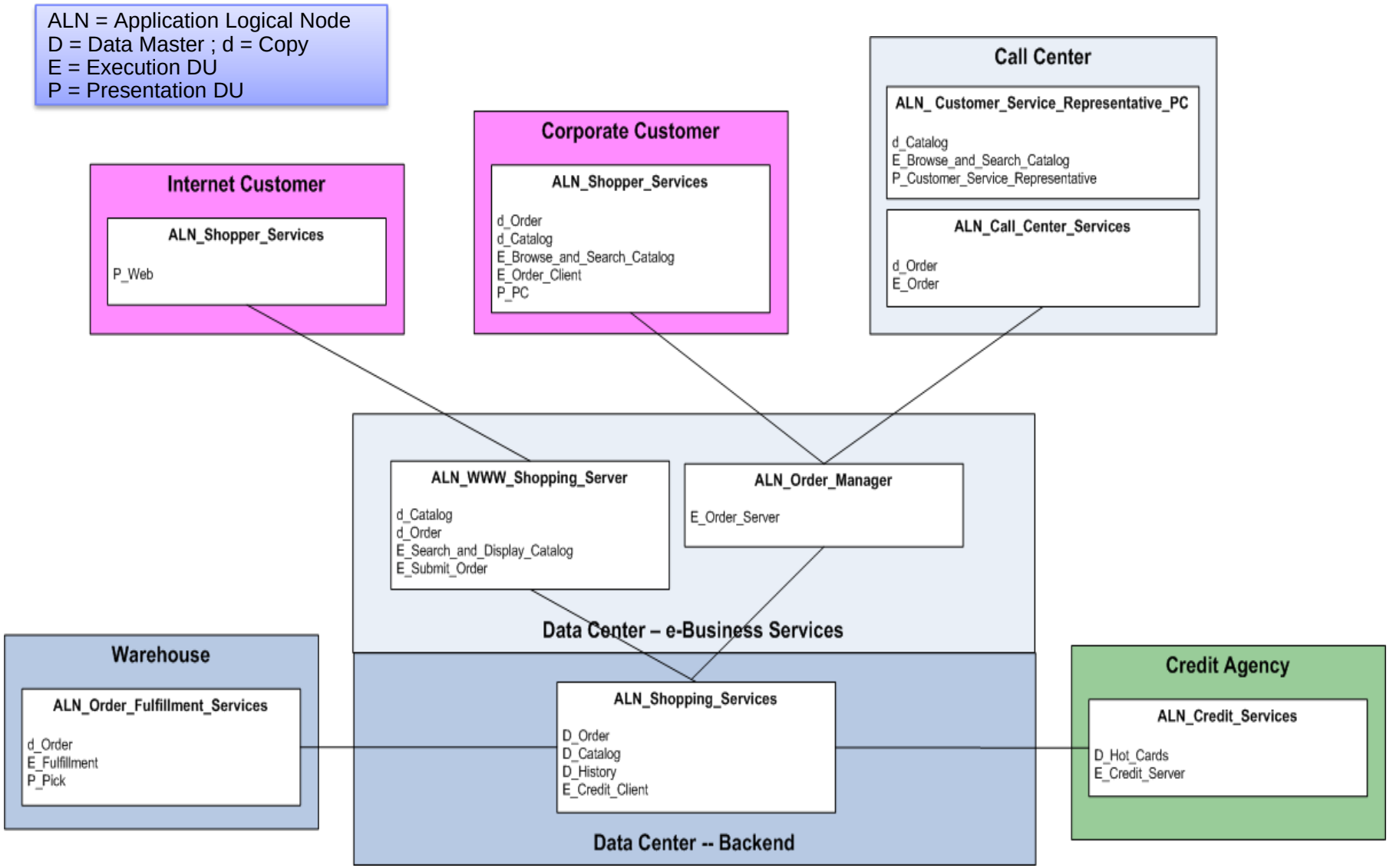




© IBM Corporation 2011

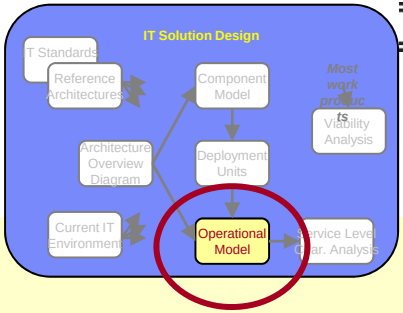
Logical Operational Model

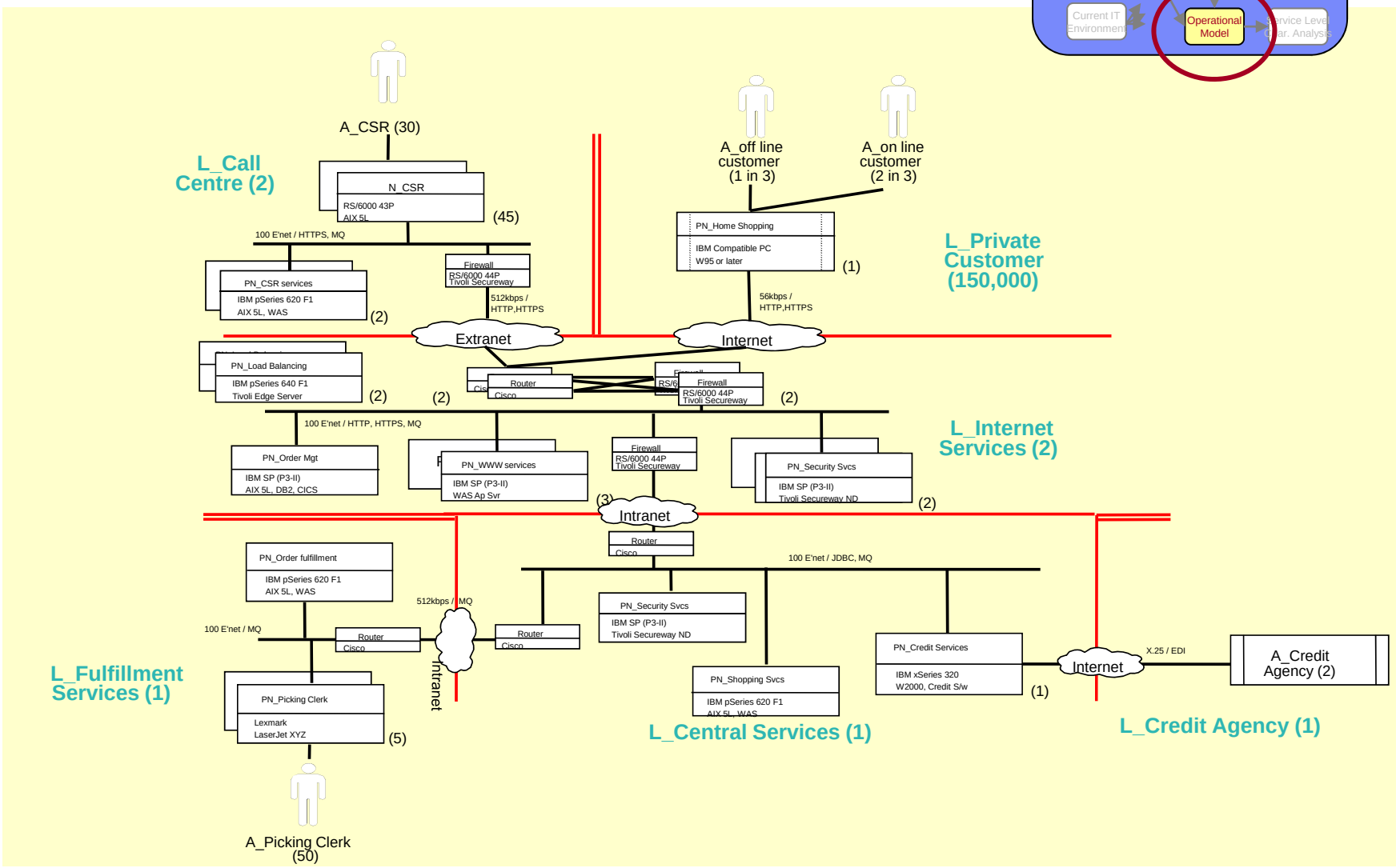
ALN = Application Logical Node
D = Data Master ; d = Copy
E = Execution DU
P = Presentation DU



© IBM Corporation 2011

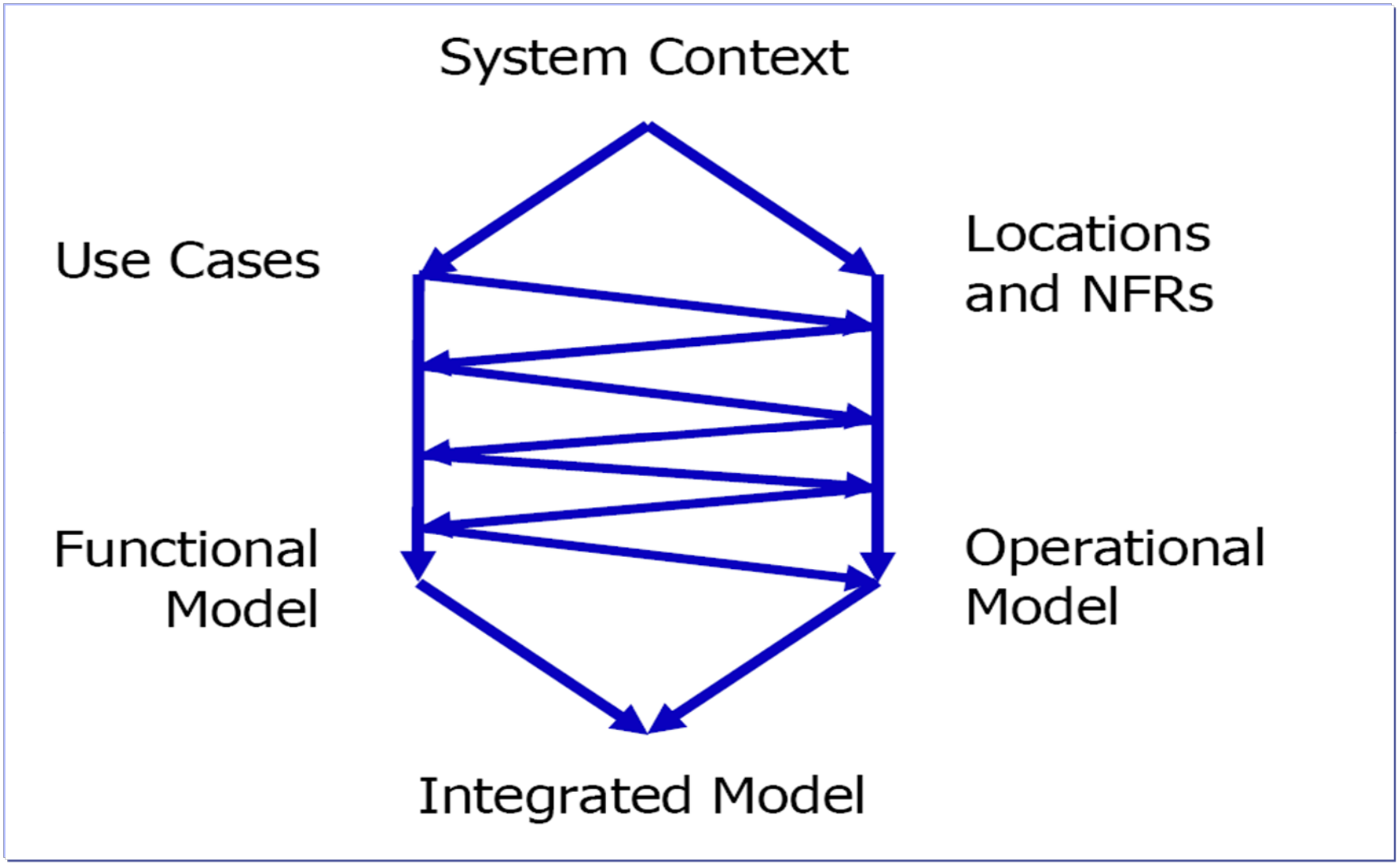
Physical Operational Model





© IBM Corporation 2011

Summary

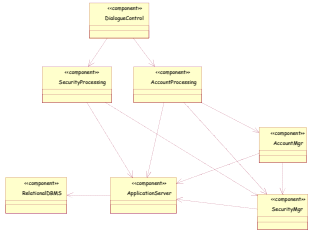


© IBM Corporation 2011

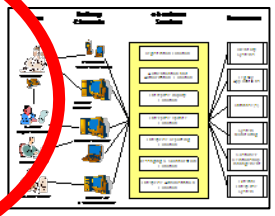
The IT architect uses four core work products to document and communicate their IT system's design

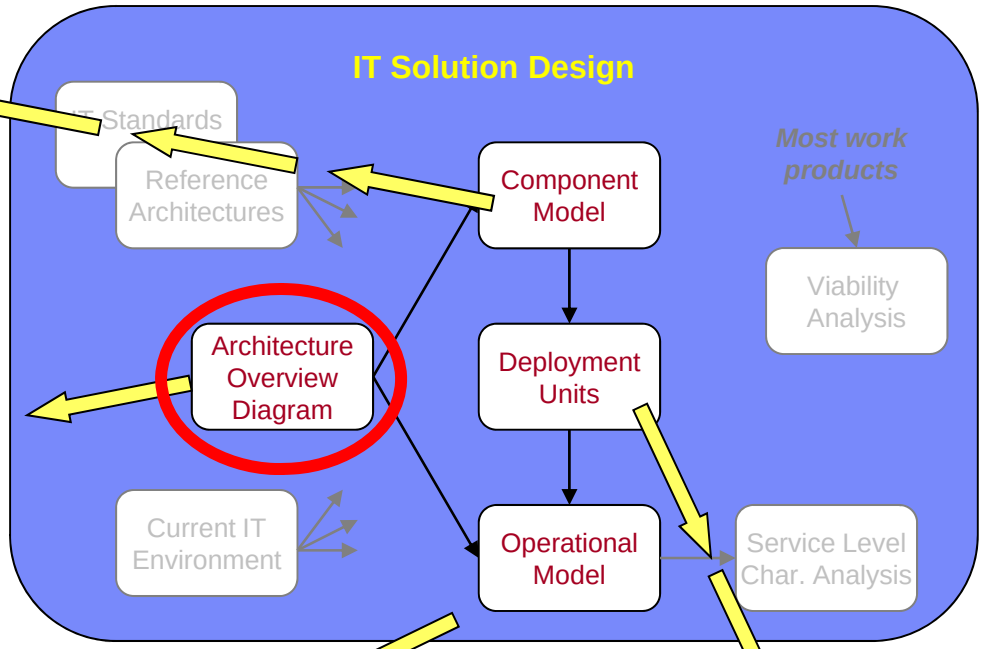
Application Architect - Software Engineering

"Component Model" describes the structure of an IT System in terms of its software components with their responsibilities, interfaces and relationships, and the way they collaborate to deliver the required functionality.

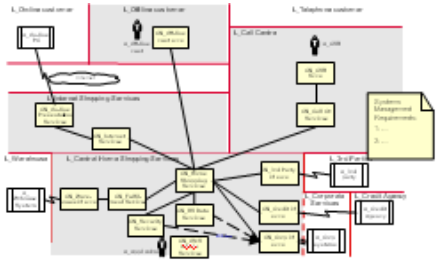


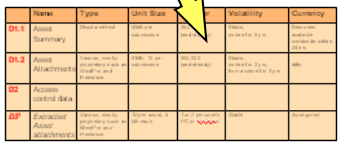
"Architecture Overview Diagram" provides a picture (not a model) of the whole IT system "on a page" as a means of communicating the salient points of the design. AODs are audience specific





"Operational Model" defines the organisation of the IT system across locations, documenting the placement of the solution's components onto nodes connected across the organisation, in order to achieve the solution's operational NFRs



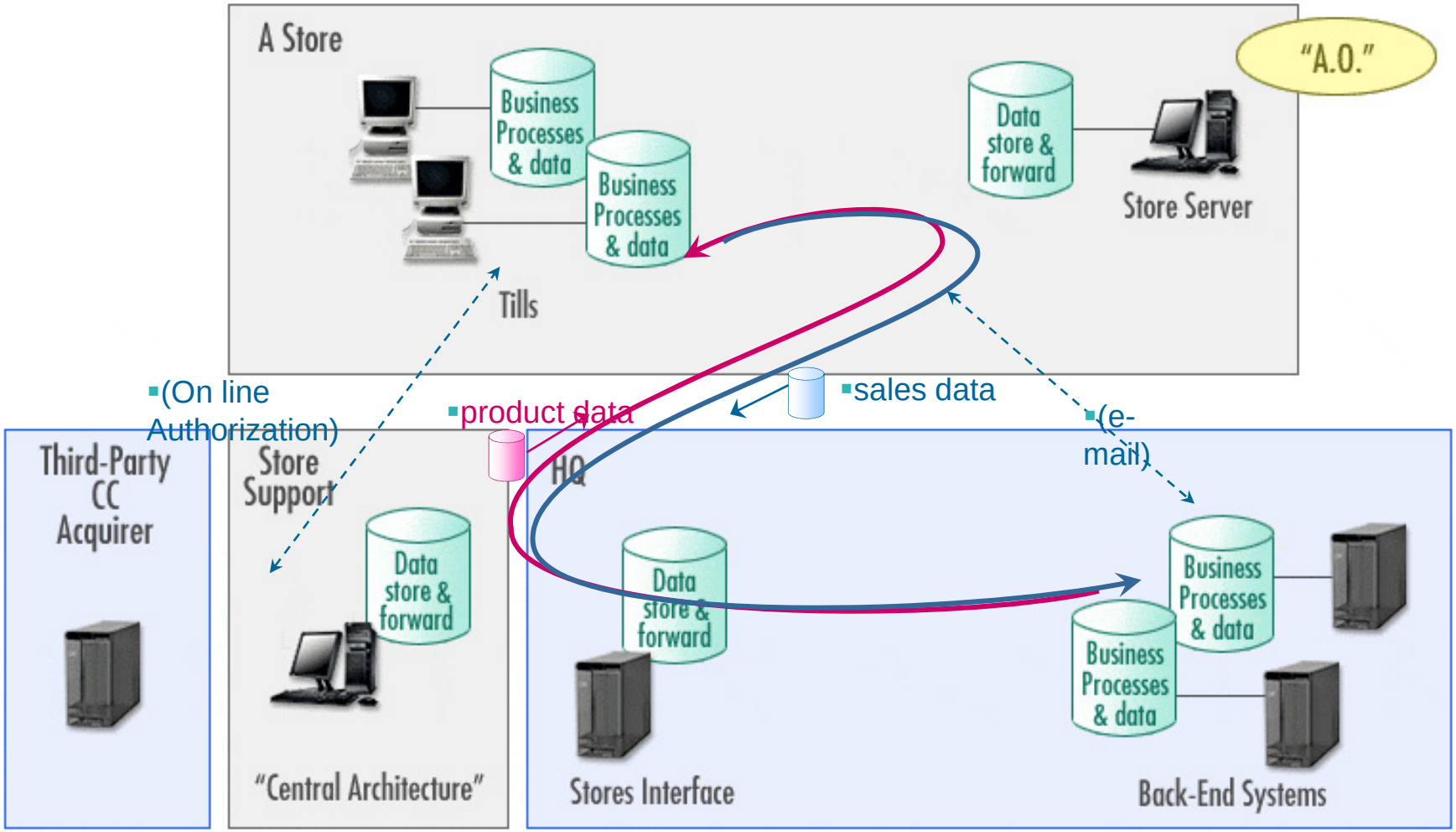


"Deployment Units" represent various aspects of components, as a convenient means of documenting their non functional requirements, as well as their placement across the Operational Model

Infrastructure Architect - Systems Engineering

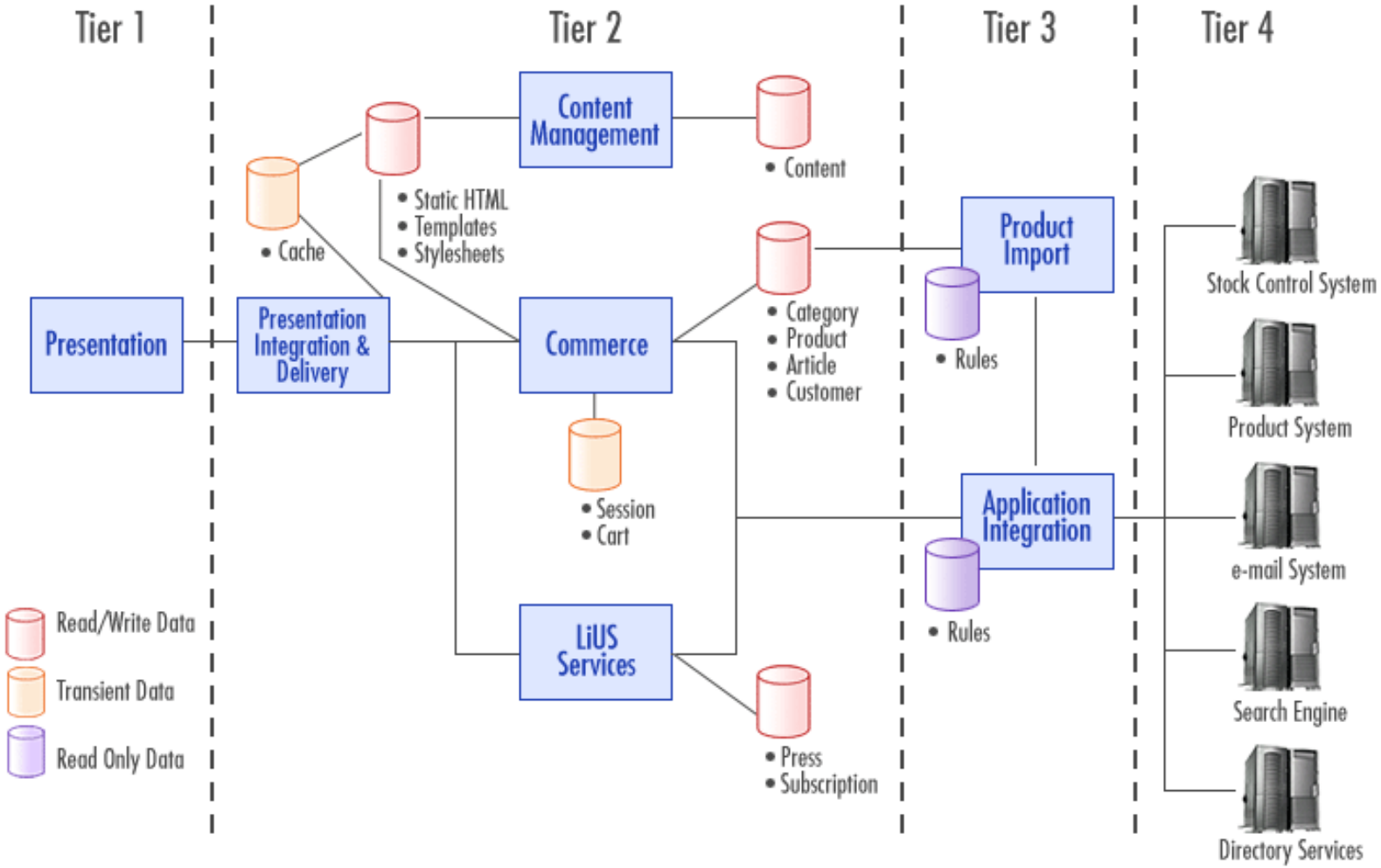
© IBM Corporation 2011

An Architecture Overview with Data Flow



© IBM Corporation 2011

An Architecture Overview showing the different tiers of a shopping system



© IBM Corporation 2011

Creating a solution architecture is an iterative process to ensure the project deals with difficulties very early in the lifecycle.

Phase Deliverable	Phase A	Phase B	Phase C
Business Requirements Specification	**Complete**	*Change control*	*Change control*
IT Solution Requirements Analysis		**Complete**	*Change control*
IT Solution Design			**Complete**

Bang!

Phase Deliverable	Solution Outline	Macro Design	Micro Design
Business Requirements Specification	High Level, qualitative	Complete	*Change control*
IT Solution Requirements Analysis	Outline System Requirements	IT scope fully defined, key NFRs	Complete
IT Solution Design	Key Requirements identified	Outline solution defined	Complete

Bang!

Catch “show-stopping problems” early in the project, enabling (if necessary) the project to be terminated at much less cost

© IBM Corporation 2011