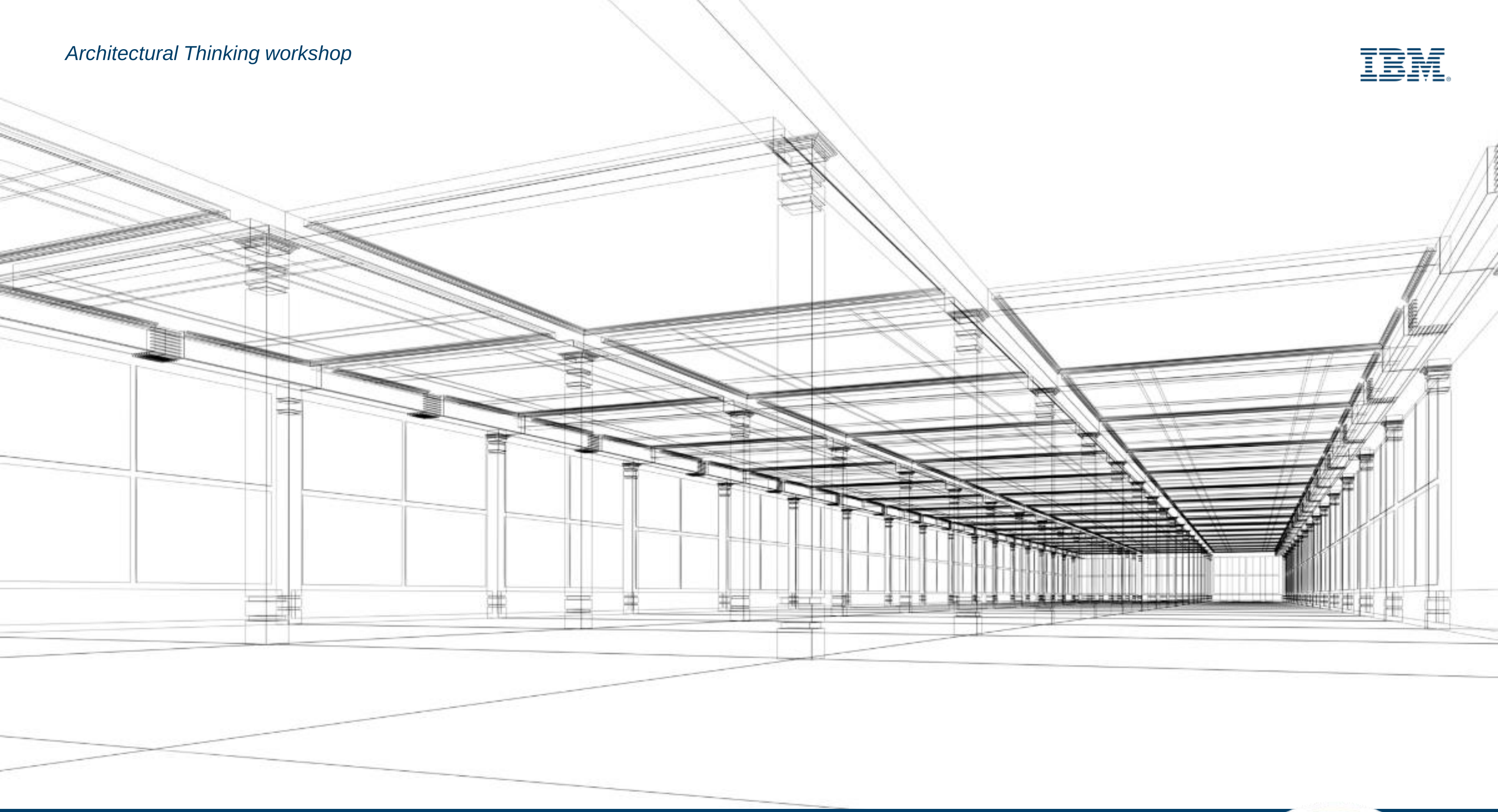


Functional Aspect

IBM Professional Series: Architectural Thinking Workshop
IBM Client Education Version 7.0

IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document m
not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

Module outline

- ▶ **Introduction and Objectives**
- The functional aspect of IT architecture
- Component modeling
- Building a component model
  - Identifying components
  - Specifying components
  - Implementing components
- Summary and references

IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

Learning objectives

At the end of this module, you should be able to:

- Describe what is meant by the functional aspect of IT architecture and how it can be modeled in a variety of ways
  - Component and Service modeling
  - Data modeling
- Describe in detail one way of modeling the functional aspect through component modeling
  - Describe what makes a “good” component
  - Describe some of the key principles used in building component models

IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

Module outline

IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

Functional Aspect with a Component Model

Why Model

- Today's complex, software-intensive systems cannot be understood without models.
- Models break down complex systems into smaller parts – components, that have well understood functional responsibilities. It helps us visualize and understand the system.

Functional aspect is one of the fundamental viewpoints to describe the system’s functional aspects

Component Model is a formal representation of:

- the internal ***Structure*** of the solution (the relationships between the components that make up the solution), i.e. *Static View****,*** and solution ***Behaviour***, i.e. *Dynamic View*.
- a next step from the System Context Diagram that aligns with the *Black Box* engineering principle to the *White Box* principle, whereby the architects look at “what’s inside the solution”
- provides the necessary input for the operational model to make proper placement decisions about the components’ various aspects:
  - Where should the components run?
  - Where should the components’ data be?
  - Where should the components be installed?
  - How will the components communicate with each other?

IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

The functional aspect of an IT system can be modeled in a variety of ways

Component and Service modelling:

- Reasoning about the structure and behavior of the system as a set of interrelated and interacting “lumps of functionality” – *Components*
- Components’ capabilities are modeled in terms of what they can do, the activities they can perform, and the information they are responsible for.
- Components work together through the exchange of information via interfaces.
- Components can manifest themselves as Services that are accessible through their interfaces or APIs that are exposed via a contract (e.g. method signature, JSON schema, WSDL, etc)

Data modelling:

- Reasoning about the structure and relationships between the system’s information entities
- Data modeling is a key part of information architecture, which also considers the informational parts of requirements, operational, and viability aspects

The focus of this module is on modeling the Functional Aspect via *Component Modeling*.

IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

The component model is used as input into a number of activities

In application development or customization:

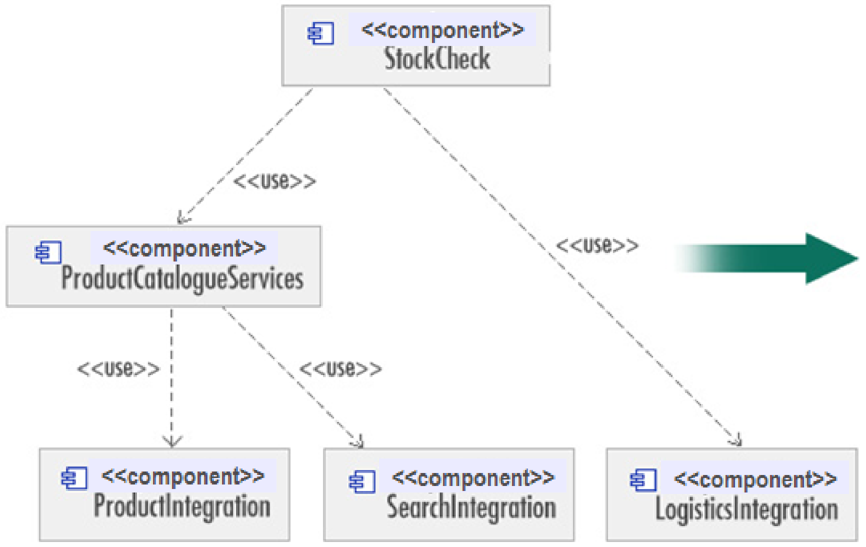
- Work allocation
- Version control
- Design strategy
- Reuse
- Testing
- Project management
- Product or Package selection

In the operational aspect:

- The *things* that have to be deployed

In operations:

- Systems management and so on

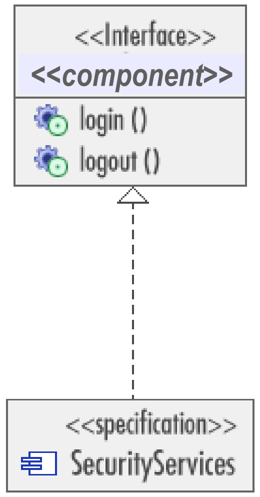

<<component>> StockCheck
<<use>>
<<use>>
<<component>> ProductCatalogueServices
<<use>>
<<use>>
<<component>> ProductIntegration
<<component>> SearchIntegration
<<component>> LogisticsIntegration

IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

A Component is a primary concept used for describing the functional aspect of the IT architecture

- A component is a modular unit of functionality.
- A component makes its functionality and state available through one or more interfaces.
- Examples of components are:
  - At the *application level* of the component model:
    - Business processing components (such as the Customer Processing component)
    - Business service components (such as the Account Manager component)
  - At the *technical level* of the component model:
    - Technical components (security services or middleware such as messaging or transaction software)
    - System software components (such as an operating system)
    - Hardware components (such as an encryption device)
- A component is not just a programming-level concept such as an Oracle® EJB or .NET component.

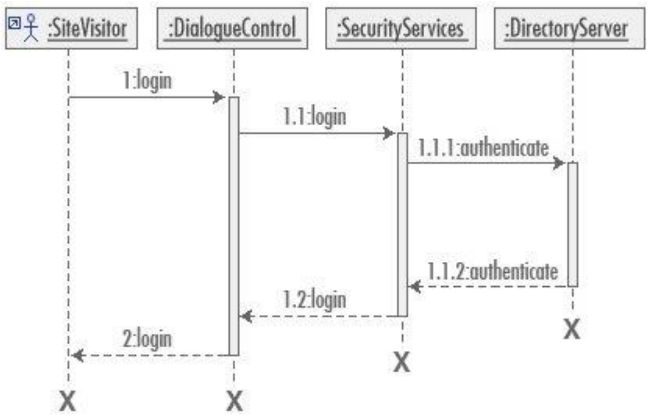
"State" – the information managed by the component



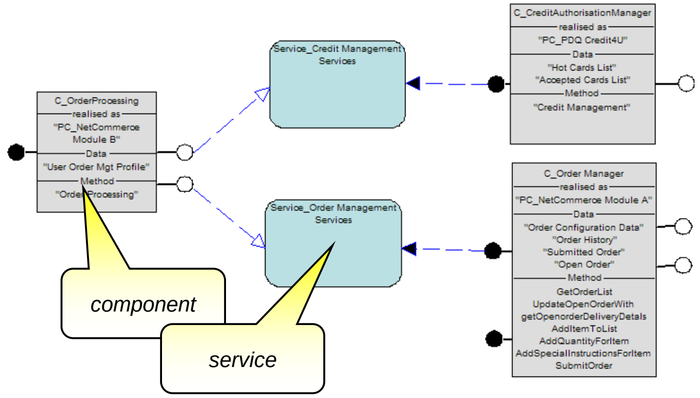
IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

Components and Services

- Components ***invoke*** each other, offering and requesting information via “messages” that flow between them.
- Using a *Black Box* engineering principle, we focus on ***what*** the components’ interfaces (or APIs) offer rather than ***how*** the requested function is realized. We still need to “look inside” components when it’s time to decide how their functions will be realized
- This approach is at the heart of *SOA* or *API Integration* architecture styles.
- *Services* and *components* are interrelated:
  - Components offer and request services via their Service Interfaces or APIs
  - Services are satisfied by components, which, if necessary, request services from other components.



Component-centric view



Service-centric view

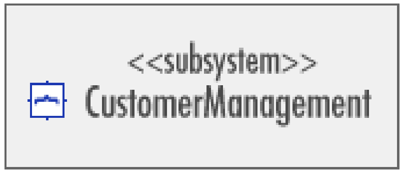
IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

Components may be grouped into subsystems

Subsystem: An interesting group of any number of any model elements

- Customer Management subsystem, Security subsystem, Printing subsystem, Order Management subsystem, Data Distribution subsystem.
- Model elements can be simultaneously in any number of subsystems.
- A subsystem has no intrinsic capabilities; it is simply an identifier.

Subsystems are commonly found in component models, where they enable the modeler to:



- Label major parts of an IT system, such as the "Accounting" subsystem
- Identify a commercial software product or package, such as CICS or SAP
- Allocate work to development groups (platform teams)

In the context of component modeling, since it is "just a label," a subsystem does not have its own interfaces. Interfaces "to the subsystem" are actually interfaces on components in the subsystem.

IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

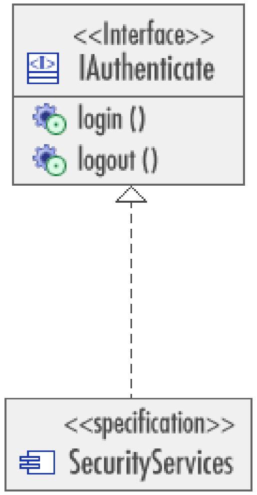
Module outline

IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

Unified Modeling Language is one of the notations used for Component Modeling

Unified Modeling Language (UML) is:

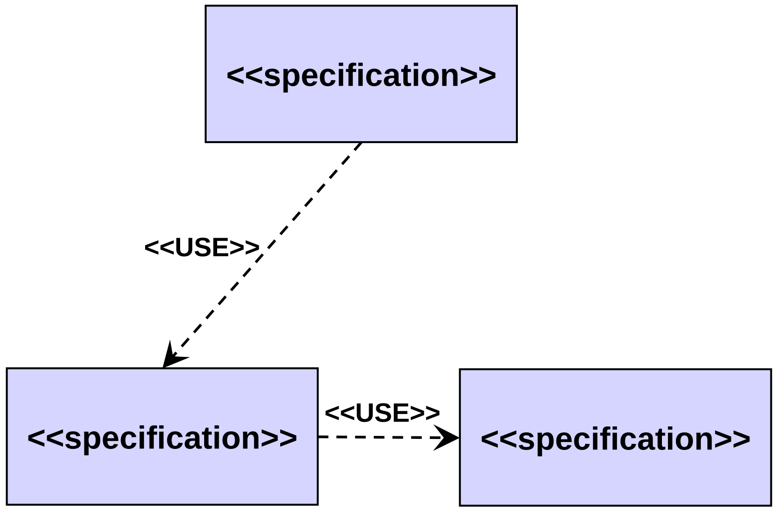
- A graphical notation, supported by a single metamodel, for describing software systems.
- An open standard controlled by the Object Management Group (OMG); Version 2.5 was released in October 2015.
- Based on a unification of the modeling work of Grady Booch, Jim Rumbaugh, and Ivar Jacobson.
- One of the most influential formal notations in software engineering and architecture, also supported by a large number of tools.



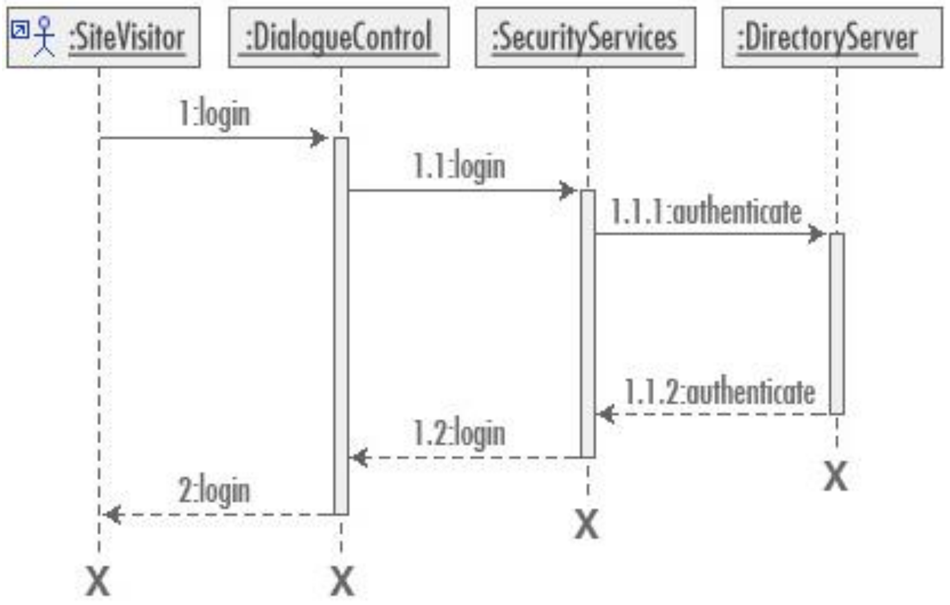
IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

Component modeling enables the functional aspect to focus …

UML Class Diagram



UML Sequence Diagram



… on the **system's internal structure**, described in a ***Component Relationship Diagram***

… as well as, **its behavior**, shown in a ***Component Interaction Diagram***

Static View

Dynamic View

IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

Components can be classified as …

Application vs Technical:

- *Application Components* implement a specific business function within the IT system (e.g. Payment Component)
- *Technical Components* are NOT associated with a specific business function. They play a technical role that supports realization of other components and supports facilitation of their non-functional requirements (e.g. a Web Server, Hypervisor, DNS Server)

Logical vs Physical

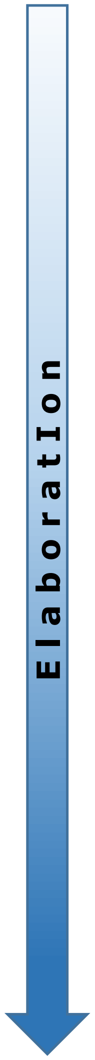
- Logical: focuses on the functional responsibility and purpose of the of the solution components without being concerned of their implementation
- Physical: Represents actual concrete software or hardware technology, a product/package used for component implementation. These are decision usually made during the Component Implementation stage (see next slide)

A component model evolves during the course of a project in a number of ways:

- Elaboration increases the level of completeness.
- Refinement increases the level of precision.
- Implementation takes place between the logical and physical levels:
  - Logical components are implemented by physical components.

IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

Three steps of component modeling technique



What are the components?

- **Component identification:**
  a. Review architectural patterns, reference architectures, and reusable assets
  b. Partition into subsystems and components and assign functional responsibilities
  c. Structure ensuring loose coupling, high cohesion, and other good qualities

What are their Interfaces?

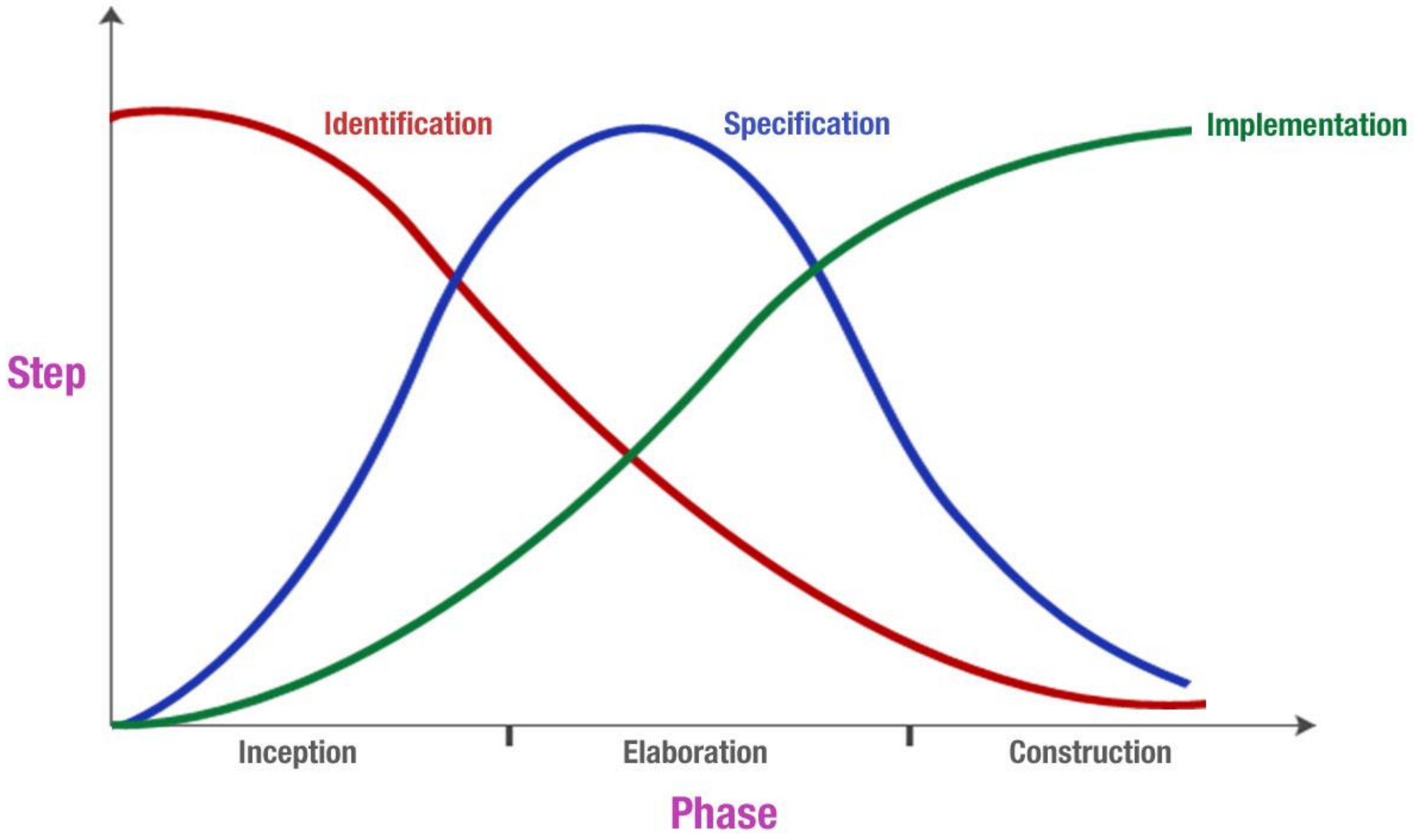
- **Component specification:**
  a. Specify interfaces
  b. Specify operations and signatures
  c. Specify pre- and post-conditions

How can we realize the components?

- **Component implementation:**
  a. Define implementation approach, make realization decisions
  b. Identify products and packages

IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

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



IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

Module outline

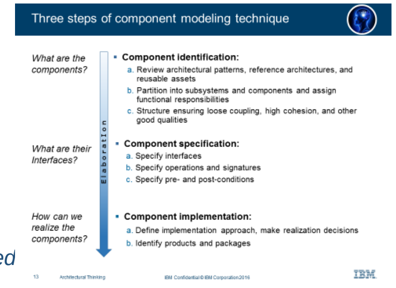
IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

Step 1: Component identification: *What are the components?*

Component identification is the first step of the component modeling technique and involves the following. Driven by an understanding of the business requirements the Architect will:

- Partition the system into subsystems and components and assign responsibilities based on an analysis of the requirements.
- Use a reference architecture, architectural patterns, and other reusable assets that can be used as a starting point
- Ensure the components are well structured, so that they are:
  - Highly cohesive
    - Do the responsibilities assigned to a component make sense?
  - Loosely coupled
    - Are two or more components too dependent on each other?
  - Well isolated
    - Have product or technology dependencies been confined to a few places?
  - Of the right granularity
    - Have sufficient responsibilities been allocated to the component?
  - Layered according to their generality

Step 1: Component Identification



Three steps of component modeling technique

What are the components?
- **Component identification:**
  a. Review architectural patterns, reference architectures, and reusable assets
  b. Partition into subsystems and components and assign functional responsibilities
  c. Structure ensuring loose coupling, high cohesion, and other good qualities

What are their Interfaces?
- **Component specification:**
  a. Specify interfaces
  b. Specify operations and signatures
  c. Specify pre- and post-conditions

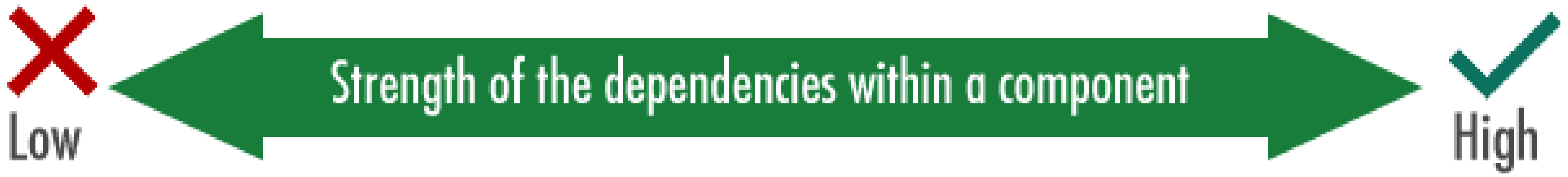
How can we realize the components?
- **Component implementation:**
  a. Define implementation approach, make realization decisions
  b. Identify products and packages

Elaboration

13 Architectural Thinking IBM Confidential © IBM Corporation 2016 IBM

IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

Cohesion is a measure of the strength of the dependencies *within* a component

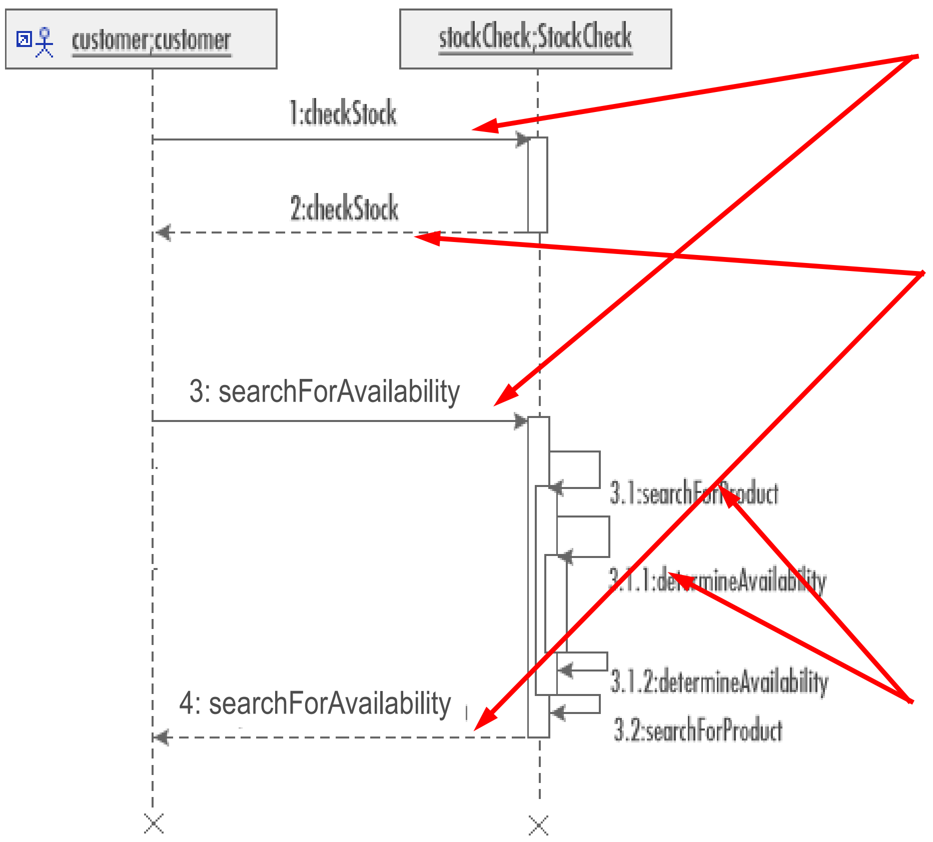


- Responsibilities have no meaningful relationship to each other.
- Responsibilities are related in time only.
- Responsibilities are of the same general category that is selected externally.

- Responsibilities contribute to a few related business activities only.
- Responsibilities are likely to use the same input or output data.
- Responsibilities are likely to be used together.

IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

Low or high cohesion? Good or bad?



Handles user input

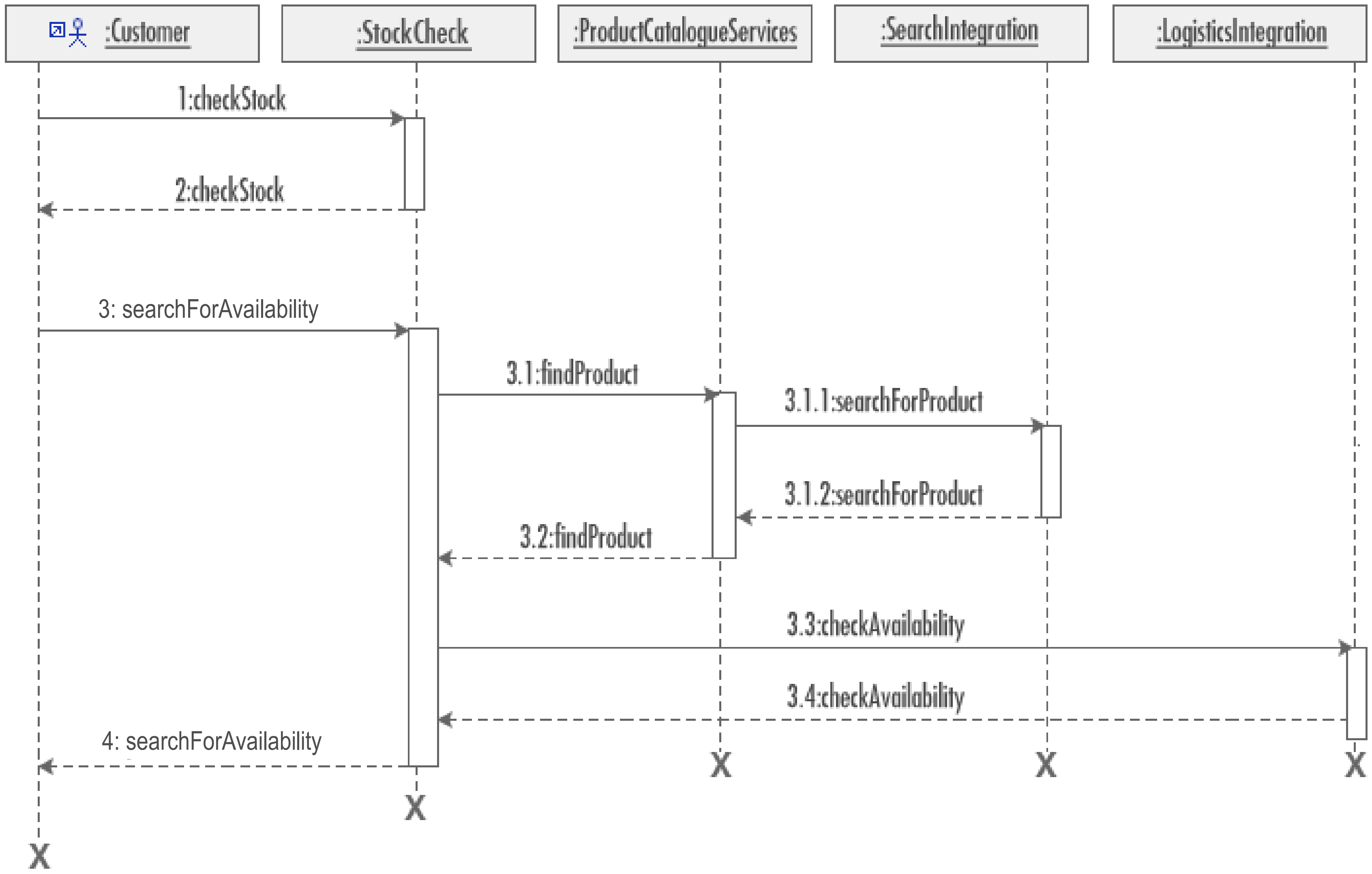
Handles display of user information

Handles different functions: search and stock availability

Handles overall control

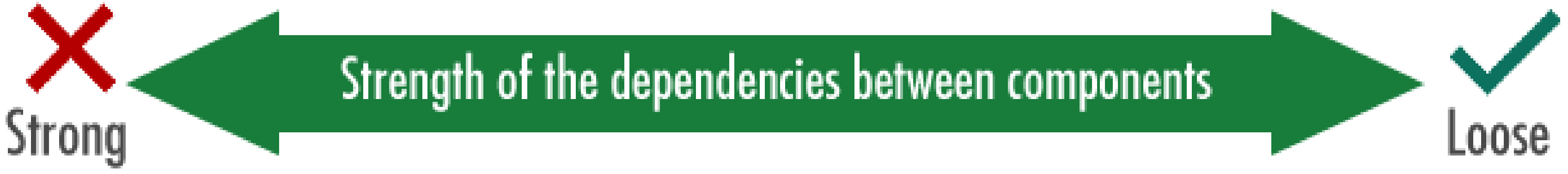
IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

An example of high cohesion: Good



IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

Coupling is a measure of the strength of the dependencies *between* components

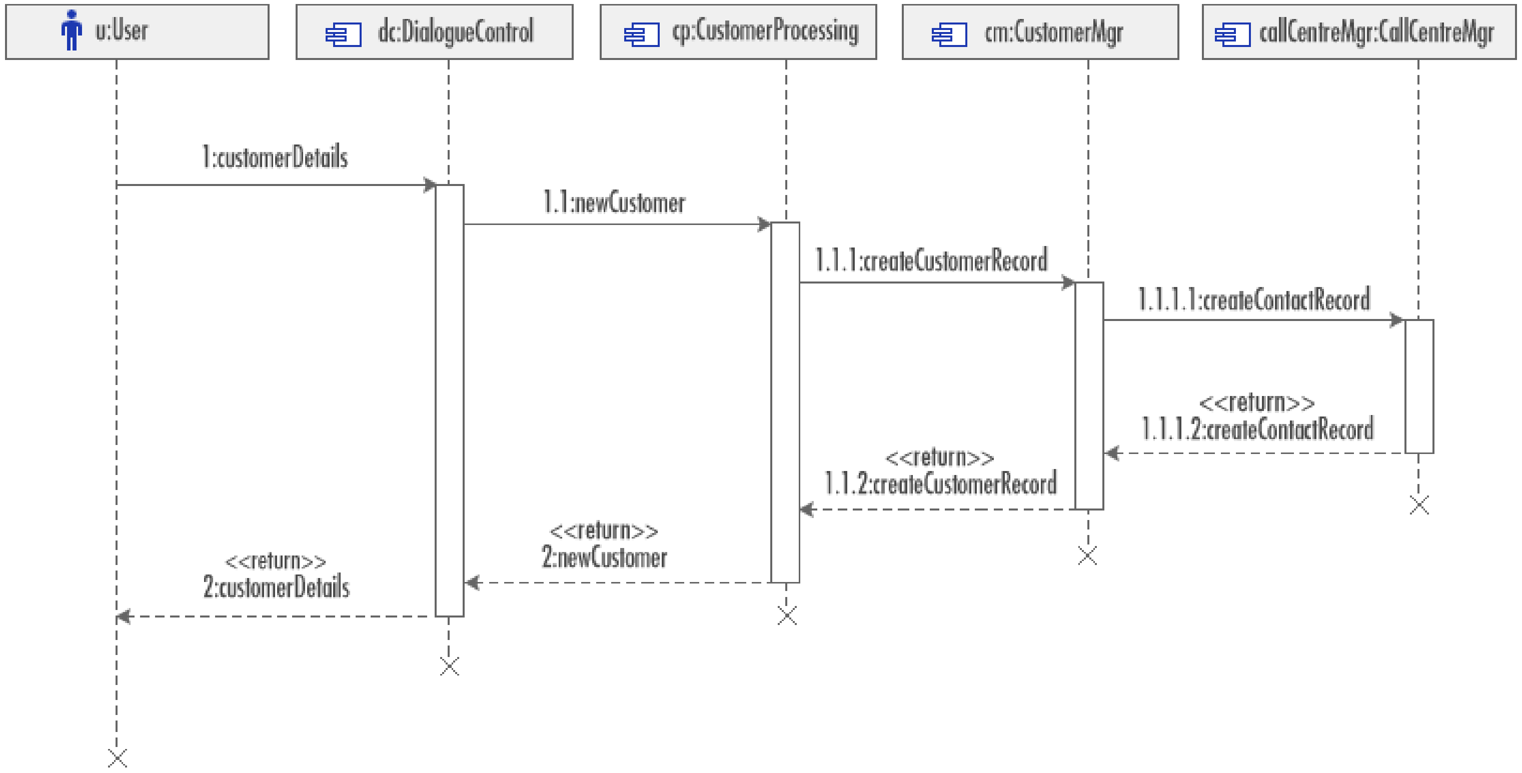


- Data that is intended to control the internal component logic is passed .
- Components refer to the same global data area.
- One component refers to the inside of another.
- Different meanings are assigned to different parts of a range of data.

- There is no unnecessary inter-component communication: Don’t Talk to Strangers.
- Communication is via fundamental data types.
- If communication is via composite data, then it is self-describing data, such as XML.

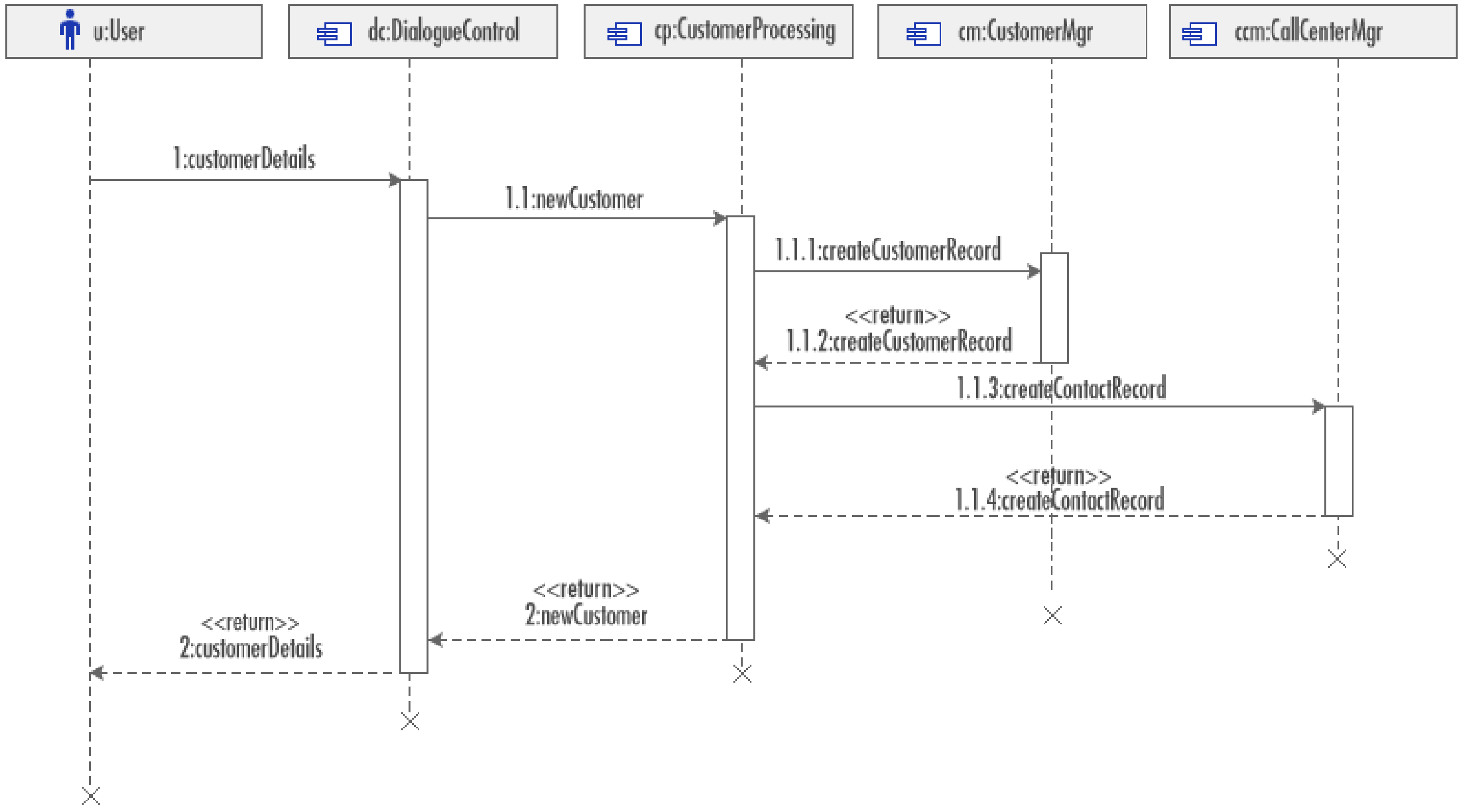
IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

An example of strong coupling: Bad



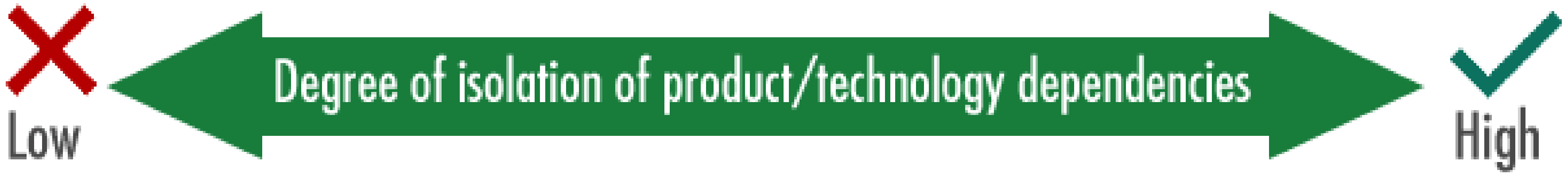
IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

An example of loose coupling: Good



IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

Isolation is a measure of the degree to which product and technology dependencies are isolated



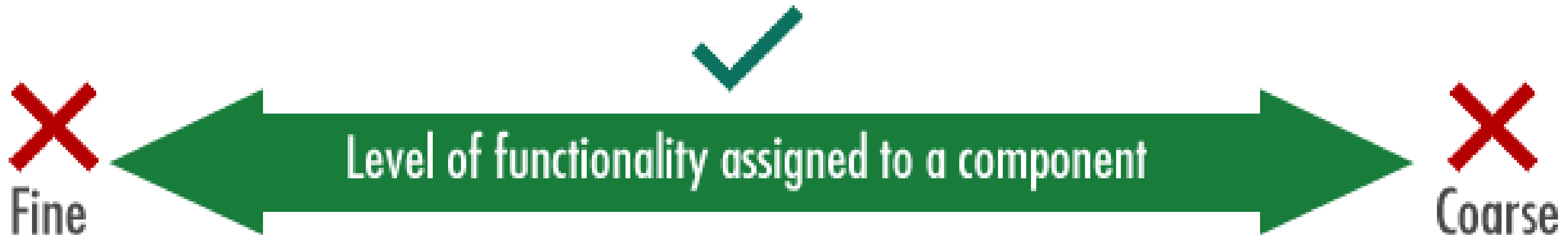
- Every component has a dependency on the product-specific or technology-specific aspects of other components.
- There is no use of patterns.

- Product and technology dependencies are decoupled.
- Patterns, such as the Gang of Four* “Proxy,” “Bridge,” and “Mediator” or GRASP** “Indirection” are used.

*See [GAMMA95]
**General Responsibility Assignment Software Patterns

IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

Granularity is a measure of the size or amount of functionality assigned to a component



- A software element called at run time with a clear interface and a separation between interface and implementation
- Also known as a *distributed component*

- An autonomous business concept or business process
- Also known as a *business component*

- A set of cooperating business components assembled to deliver a solution to a business problem
- Also known as a *system-level component*

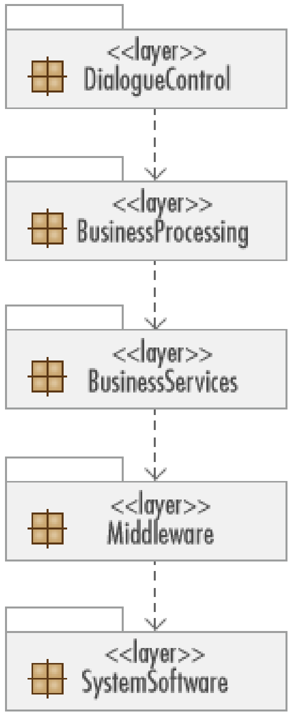
IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

Layering involves separating out components according to their generality

- Layering provides a logical partitioning of components into a number of sets or 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.

Example:

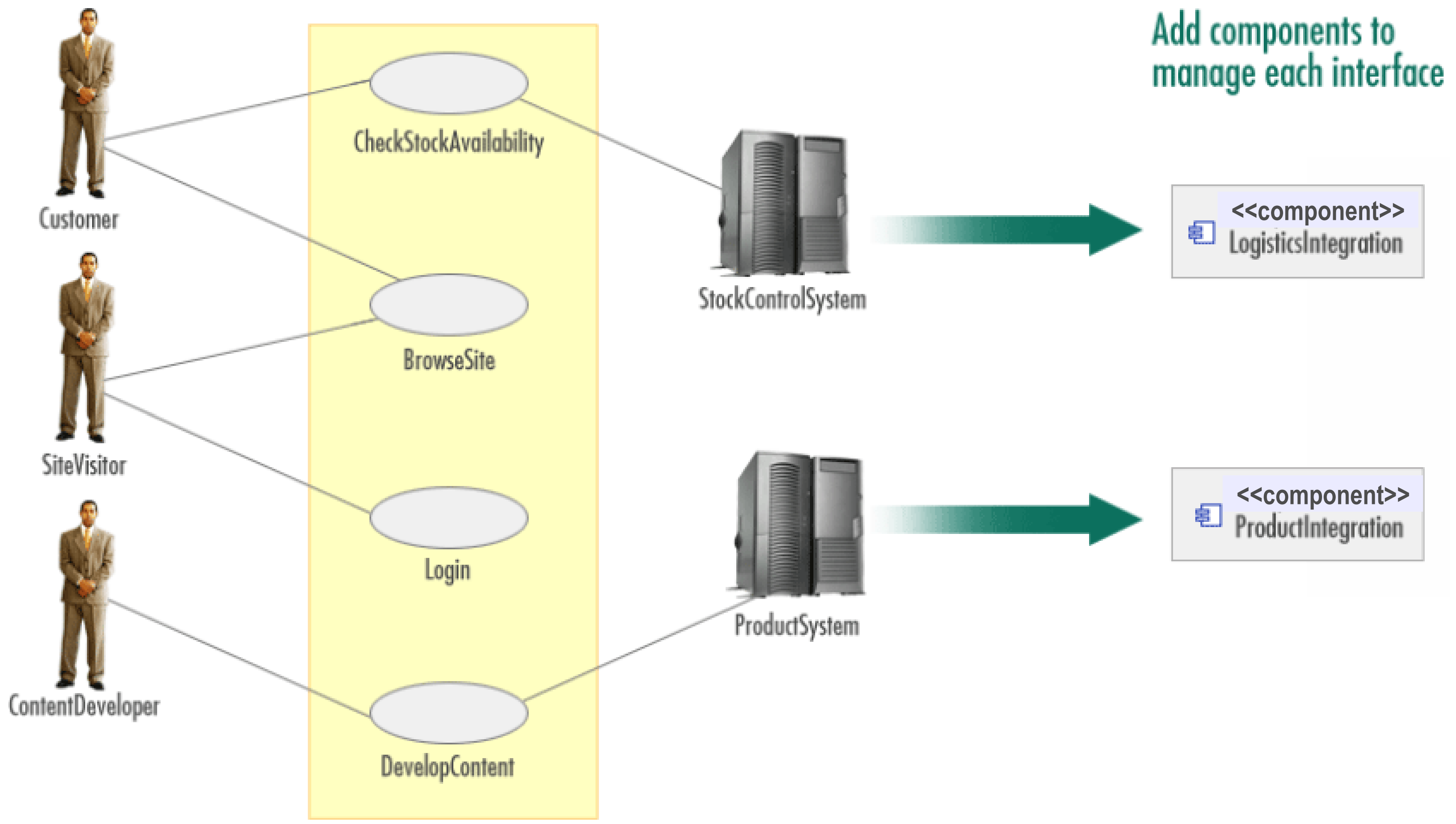
- Dialog control layer
- Business processing layer - use case step logic and choreography.
- Business services layer - general business components, may be used in several applications.
- Middleware layer - interfaces to databases and operating system services.
- System software layer - components of operating systems and databases.


<<layer>> DialogueControl
<<layer>> BusinessProcessing
<<layer>> BusinessServices
<<layer>> Middleware
<<layer>> SystemSoftware

IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

Partitioning

Partition into subsystems and components and assign responsibilities; identify interfaces to external systems.



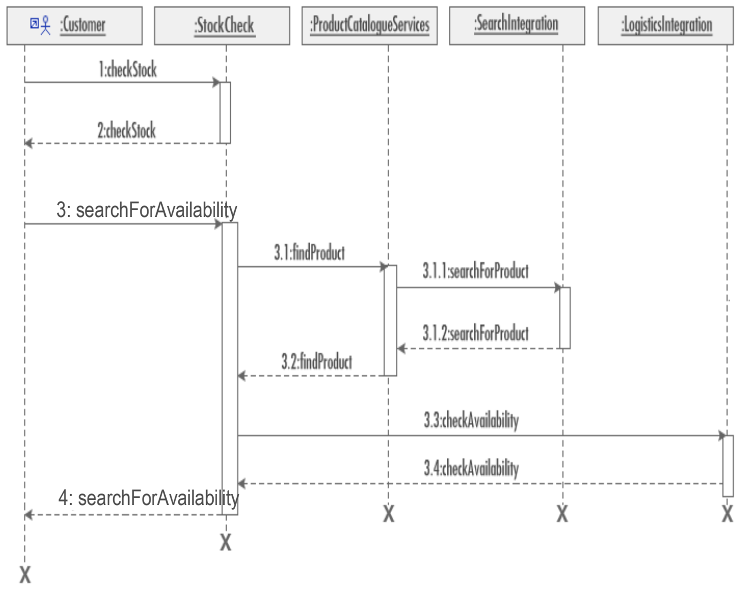
IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

Identifying components for architecturally significant use cases

Partition into subsystems and components and assign responsibilities; identifying components for architecturally significant use cases.

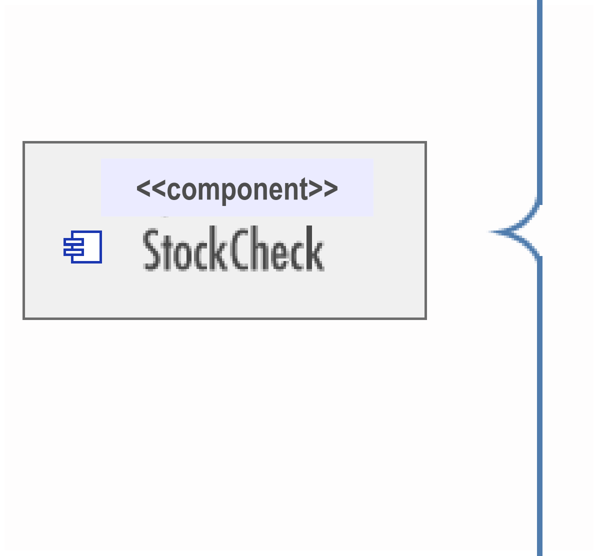
- The actor requests a stock query.
- The system prompts the actor to select a store name and enter an article or product name or number.
- The actor selects a store name and enters an article or product name. If a product name is entered, the user may also be required to specify further selection criteria, such as selecting color and so on, for that product.
- The system searches for the entered article or product in the selected store and checks its availability.
- The system determines that the article or product is available.
- The system returns the availability information for the article or product.
- The use case ends successfully.

Use Case: Check stock availability



IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

Partition into subsystems and components and assign responsibilities to component

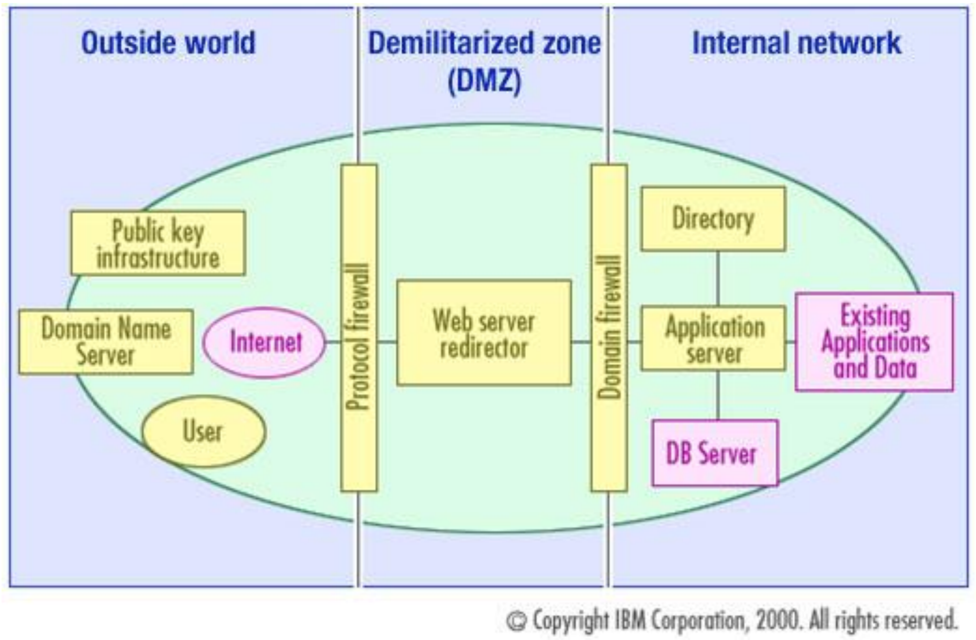


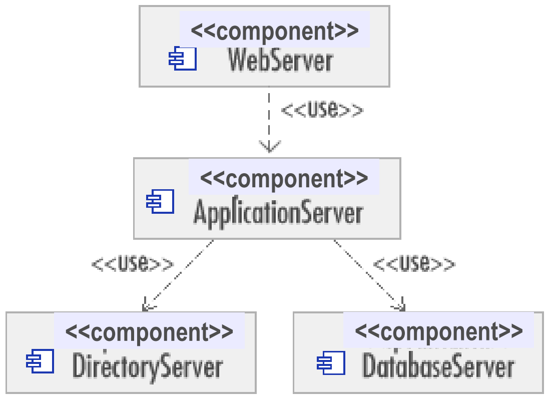
- Handle stock requests
- Request product information
- Perform search
- Determine availability
- Provide availability information

IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

Review architectural patterns, reference architectures, and reusable assets

Review architectural patterns, reference architectures, and reusable assets to derive an initial set of components





<<component>> WebServer
<<use>>
<<component>> ApplicationServer
<<use>>
<<use>>
<<component>> DirectoryServer
<<component>> DatabaseServer

IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

The completion of the component identification step

At the completion of the component identification step, there should be:

- An initial set of components with associated responsibilities. This may be documented as a Component or Service Catalog
- One or more component relationship diagrams showing dependencies between components
- One or more component interaction diagrams showing collaborations between components
- Subsequent steps will:
  - (2) Specify components: interfaces, operations, and signatures
  - (3) Show how logical components are implemented into physical components

IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

Module outline

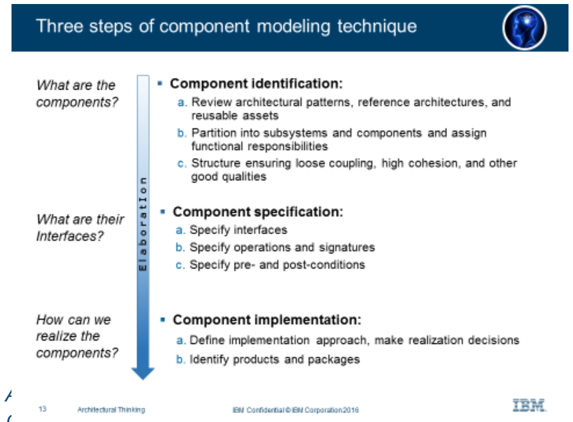
IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

Step 2: Component specification: *What are the Interfaces?*

Component specification is the second step of the component modeling technique and involves:

- Defining component interfaces by separating out the responsibilities placed on the components
- For each interface, specifying the operations and their signatures, that is, parameters passed and return values
- For each operation, specifying its contract of behavior or pre- and post-conditions

Step 2: Component Specification



Three steps of component modeling technique

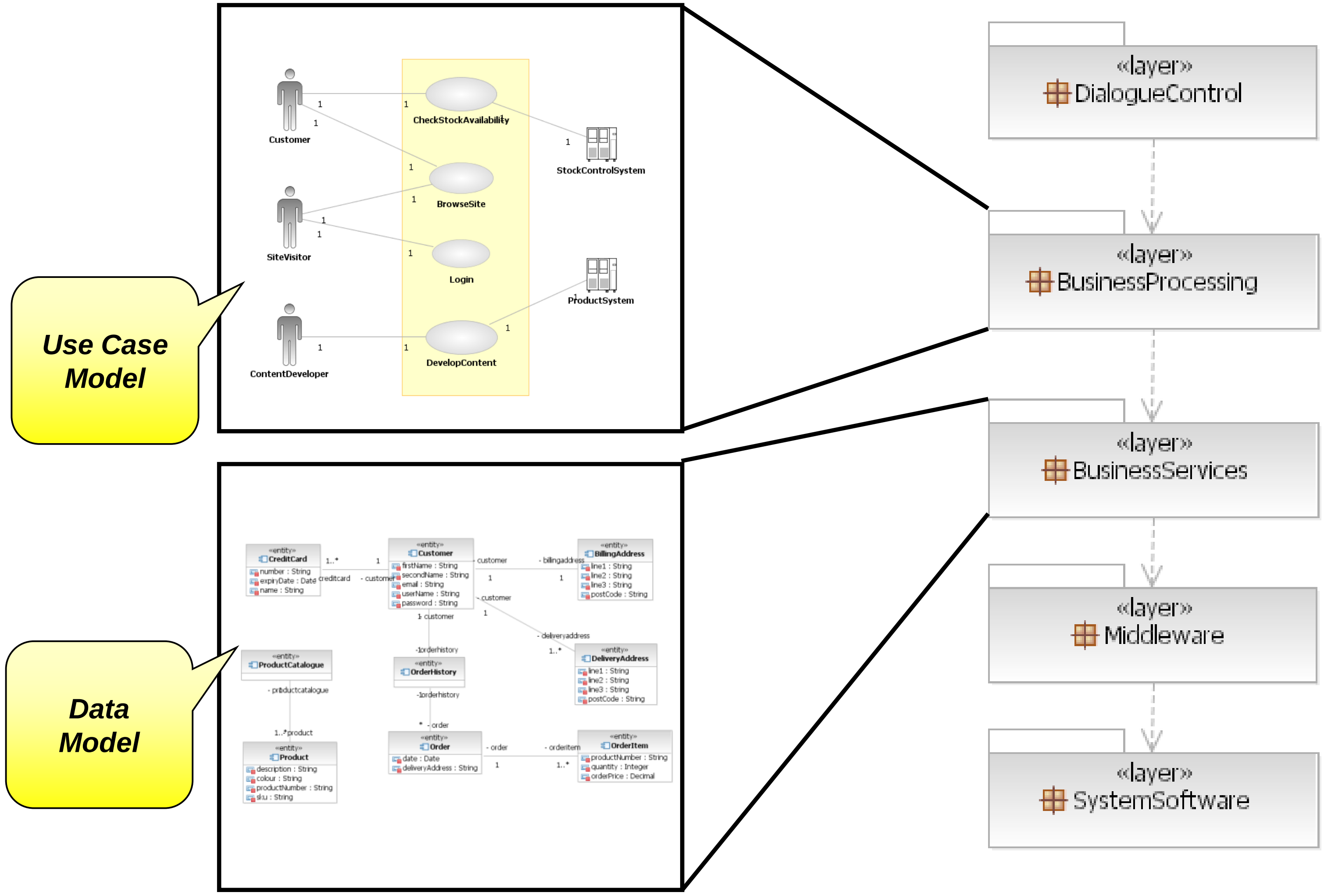
What are the components?	▪ **Component identification:** a. Review architectural patterns, reference architectures, and reusable assets b. Partition into subsystems and components and assign functional responsibilities c. Structure ensuring loose coupling, high cohesion, and other good qualities
What are their Interfaces?	▪ **Component specification:** a. Specify interfaces b. Specify operations and signatures c. Specify pre- and post-conditions
How can we realize the components?	▪ **Component implementation:** a. Define implementation approach, make realization decisions b. Identify products and packages

Elaboration

13 Architectural Thinking IBM Confidential © IBM Corporation 2016 IBM

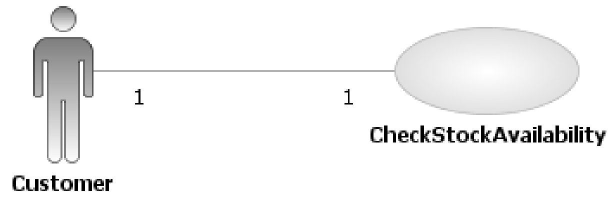
IBM Confidential. © Copyright International Business Machines Corporation 2017. A
not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

There are various approaches to identifying interfaces on IT components, depending on which layer they are in

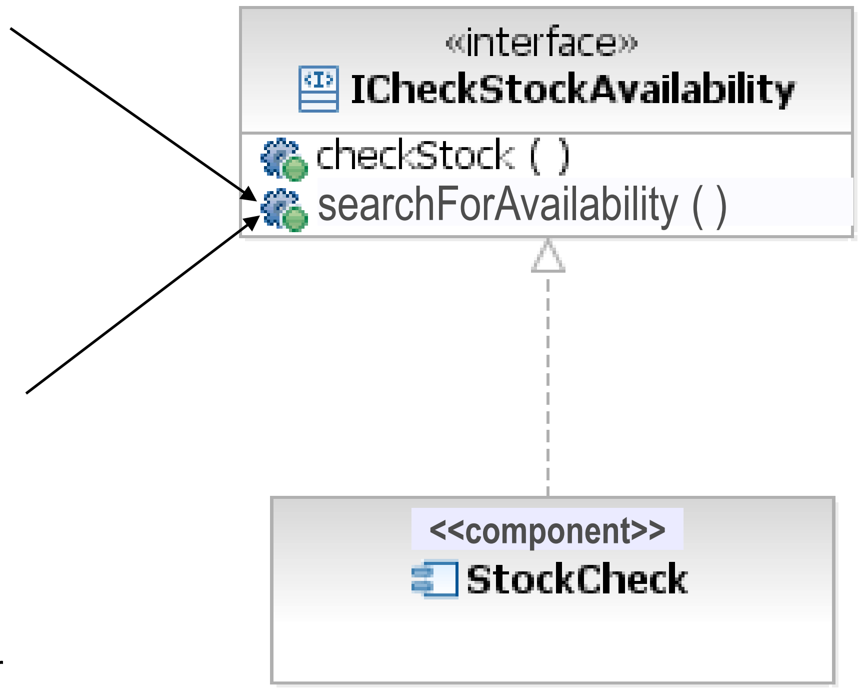


IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

Identifying business processing component interfaces from use cases



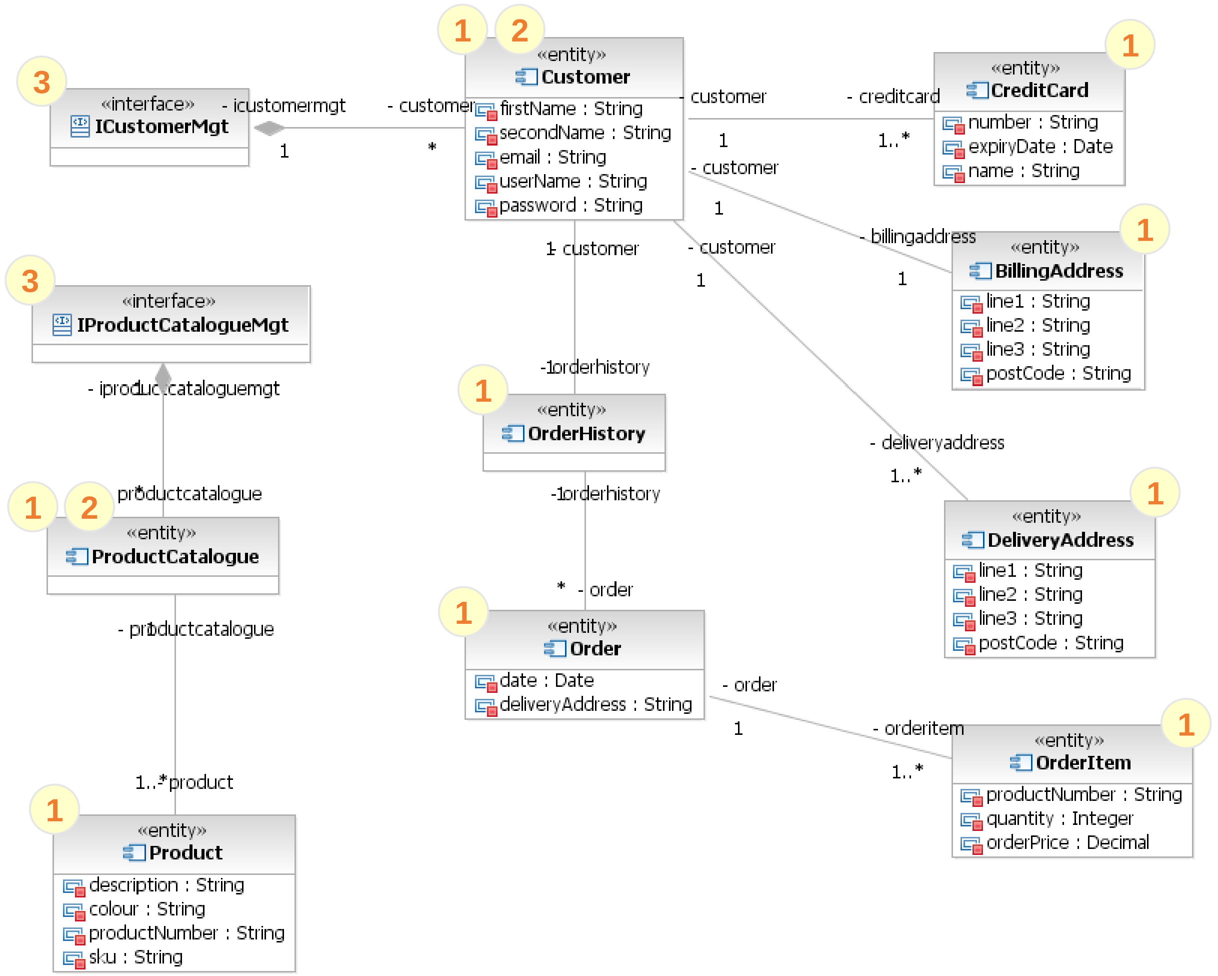
1. The actor requests a stock query.
2. The system prompts for the actor to select a store name and enter an article or product name or number.
3. The actor selects a store name and enters an article or product name. If a product name is entered, the user may also be required to specify further selection criteria (for example, selecting color) for that product.
4. The system searches for the selected article or product in the named store and checks its availability.
5. The system determines the article or product is available.
6. The system returns the availability information for the article or product.
7. The use case ends successfully.



IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

Identifying business service component interfaces from business entities

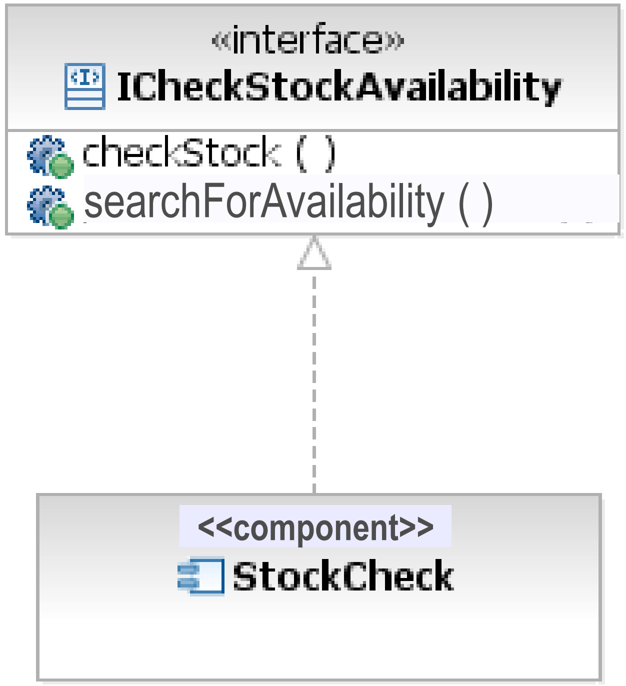
1. Produce a logical data model showing business entities.
2. Identify core business entities.
3. Create interfaces and operations to manage core business entities.



IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

Assigning operation signatures to operations

- For each operation identified, decide what data flows in and out. Make decisions about what data is required to provide the required functionality.
- Look at the collaborations between components in component interaction diagrams.
- Create structured data types as appropriate. Later in the cycle, we will use the modelling structures aligned with the implementation technology (e.g. method signature, JSON schema, WSDL, etc)



Let’s look at the interaction **searchForAvailability**:

- A store name and a product name, both strings, are required as input.
- The quantity available in the store will be returned.
- The signature then becomes:

searchForAvailability (storeName:String, product:String):Integer

IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

The completion of the component specification step

There should be a refined component model with:

Defined component interfaces by separating out the responsibilities placed on the components:

- For each interface, specifying the operations and their signatures, that is, parameters passed and return values
- For each operation, specifying its contract of behavior or pre- and post-conditions

Subsequent step will:

- Map the specified components to the appropriate products, packages, and custom-built applications; an architecture decision needs to be taken
- Identify implementation approach of physical components

IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

Module outline

Introduction and objectives

The functional aspect of IT architecture

Component modeling

▶ **Building a component model**

- Identifying components
- Specifying components
- ▶ **Implementing components**

Summary and references

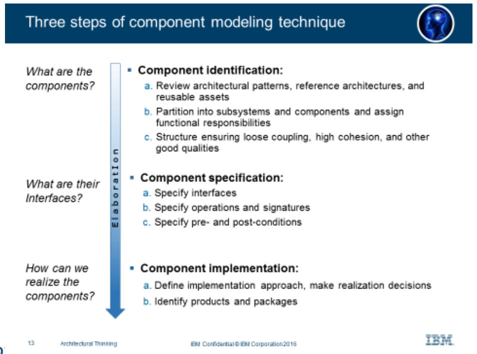
IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

Step 3: Component Implementation: *How can we realize the Components?*

Component Implementation is the third step of the component modeling technique and involves:

- For those components that are to be implemented using commercial-off-the-shelf (COTS) products, map the specified-level components to the appropriate products or packages.
- For those components that are to be built, identify the approach to implementation by defining the physical-level components. Use patterns or other assets to help in this definition.
- And don't forget that many components will be realized as standard technologies defined with selection criteria in the EA.
  - Whether at the *IS* level of IT business functionality
  - Or the *technology* level of IT middleware and so on
- Enterprise Architectures frequently adopt the *Reuse before Customise before Buy before Build* guideline.

Step 3: Component Implementation



Three steps of component modeling technique

What are the components?
- **Component identification:**
  a. Review architectural patterns, reference architectures, and reusable assets
  b. Partition into subsystems and components and assign functional responsibilities
  c. Structure ensuring loose coupling, high cohesion, and other good qualities

What are their Interfaces?
- **Component specification:**
  a. Specify interfaces
  b. Specify operations and signatures
  c. Specify pre- and post-conditions

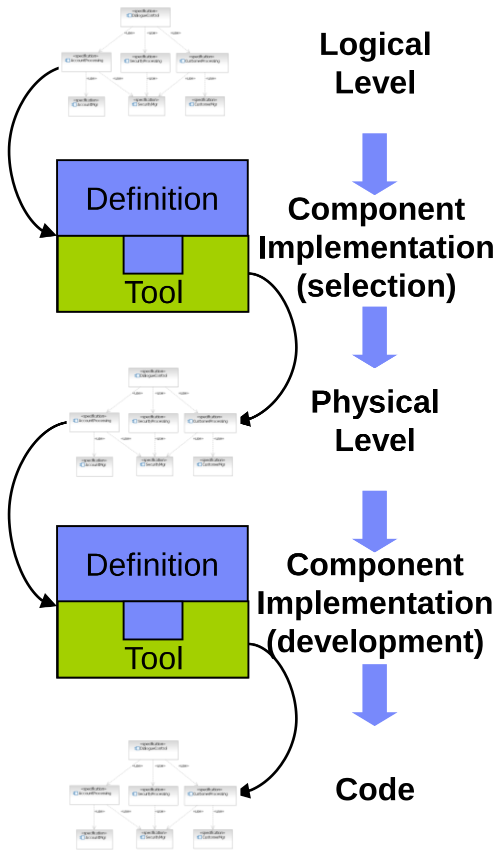
How can we realize the components?
- **Component implementation:**
  a. Define implementation approach, make realization decisions
  b. Identify products and packages

IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

Transformation from a logical-level model to a physical-level model

During component implementation, the logical-level model is transformed into a physical-level model

- Decisions must be made as to what technology best realizes the specified-level components.
- Implementing a specified-level component model into a physical-level component model involves:
  - Selecting products and packages
  - Identifying frameworks and patterns
  - Deciding what components have to be developed
  - Making choices about what technology should be used to realize the components
- There can be more than one physical component model for each specified-level model.
- Strictly speaking, this notion of component implementation is implementation selection, to be followed by application development or package customization.


Logical Level
Definition
Tool
Component Implementation (selection)
Physical Level
Definition
Tool
Component Implementation (development)
Code

IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

Module outline

IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

Summary

The **Functional Aspect** of a system’s IT architecture use Component Modeling techniques to define the models.

Component Modeling describes the functional structure and behavior of the system’s IT architecture via components.

- Components’ capabilities are modeled in terms of what they can do, the activities they can perform and the information they are responsible for.
- Components work together through the exchange of messages sent and received via interfaces.

Component models can include:

- **Static**, structural views and **dynamic**, behavioral views
- Application and technical levels
- Logical and physical perspectives

Component modeling consists of three steps:

- Component Identification
- Component Specification
- Component Implementation

IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

References

- **[BACHMANN00]** Bachmann, et al. *Software Architecture Documentation in Practice: Documenting Architectural Layers*, SEI, March 2000
- **[BOOCH93]** Booch, Grady. *Object Oriented Analysis and Design with Applications*, Benjamin Cummings, 1993, ISBN: 0805353402
- **[CHEESMAN00]** Cheesman, John and Daniels, John. *UML Components*, Addison-Wesley, 2000, ISBN: 0201708510
- **[GAMMA95]** Gamma, Erich, et al. *Design Patterns: Elements of Reusable Object Oriented Software*, Addison-Wesley, 1995, ISBN: 0201633612
- **[HERZUM99]** Herzum, Peter and Sims, Oliver. *Business Component Factory - A Comprehensive Overview of Component-based Development for the Enterprise*, John Wiley, 1999, ISBN: 0471327603
- **[JACONSON94]** Jacobsen, Ivar, et al. *Object Oriented Software Engineering: A Use Case Driven Approach*, Addison Wesley, 1994, ISBN: 0201544350
- **[MEYER97]** Meyer, Bertrend. *Object-Oriented Software Construction (2nd Edition)*, Prentice-Hall, 1997, ISBN: 0136291554
- **[PAGE-JONES88]** Page-Jones, Meiler. *The Practical Guide to Structured Systems Design*, Prentice Hall, 1988, ISBN: 0136907776
- **[WIRFS-BROCK90]** Wirfs-Brock, Rebecca, et al. *Designing Object Oriented Software*, Prentice Hall, 1990, ISBN: 0136298257

IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.

THANK YOU

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this infomation with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information." Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and its affiliates.

Linux is a trademark of Linus Torvalds in the United States, or other countries, or both. Microsoft, Excel, PowerPoint, Project, Visio, Word, and Windows are trademarks of the Microsoft Corporation in the United States, other countries, or both. UML is a registered trademark of Object Management Group, Inc. in the United States and/or other countries. TOGAF and UNIX are registered trademarks of The Open Group in the United States and other countries. Other company, product, and service names may be trademarks of IBM or other companies. Citrix is a trademark of Citrix Systems, Inc. and one or more of its subsidiaries, and may be registered in the United States Patent and Trademark Office and in other countries.

IBM Confidential. © Copyright International Business Machines Corporation 2017. All Rights Reserved. This document may not be copied distributed or reproduced in whole or in part without the prior written consent of IBM.