Software Service Engineering

Prof. Dr.-Ing. Martin Gaedke
Technische Universität Chemnitz
Fakultät für Informatik
Professur Verteilte und selbstorganisierende Rechnersysteme

http://vsr.informatik.tu-chemnitz.de

Section 2

DEVELOPING SOFTWARE IN A CONNECTED WORLD

Web Computing Introduction

- Programming with distributed functionalities on the Web
  - Calling interfaces / endpoints
  - Heterogeneous, distributed, multi-language environment
- Different approaches exists to wire distributed components
  - Wrt.: Wire Protocol, Wire Formats
  - Message passing, RPC, Web Service/SOAP

Message Passing Model

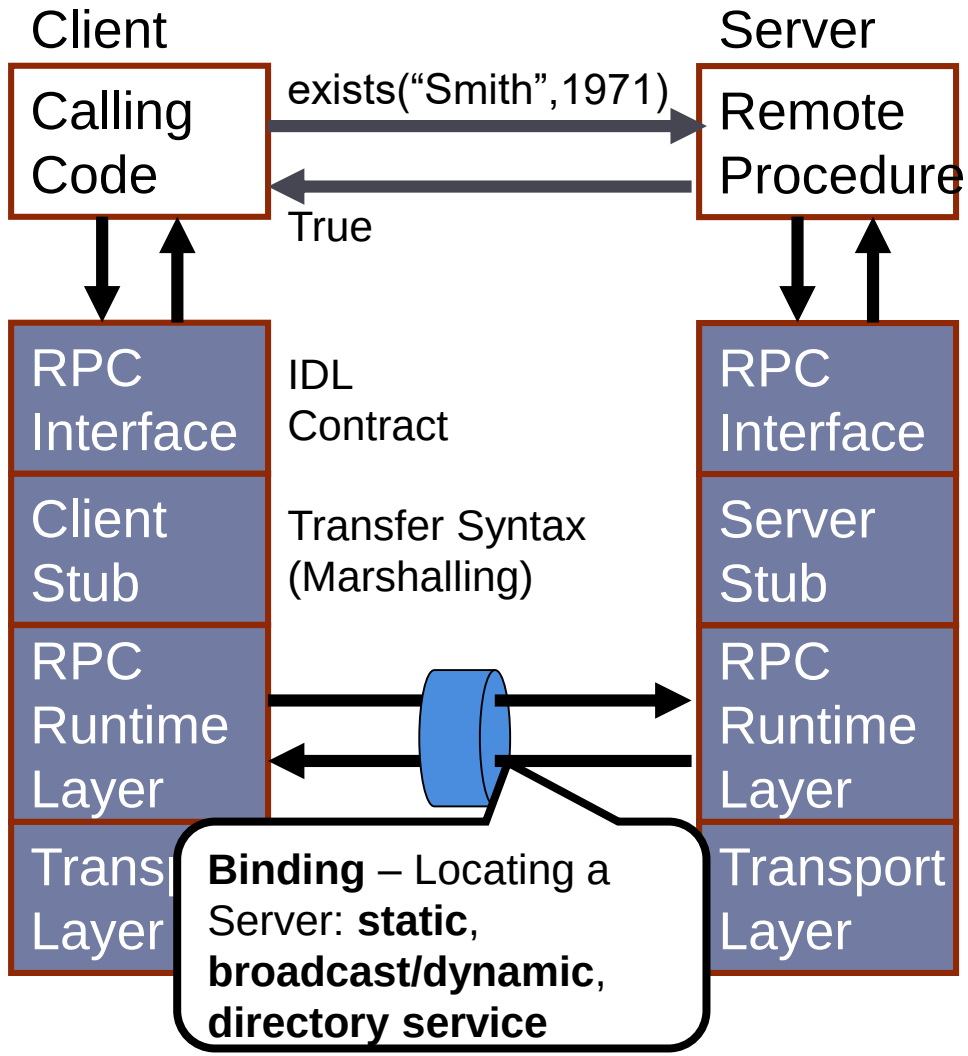
- Sender-Receiver Paradigm
  - **Message**: (Typified) Data transmitted from sender (S) to receiver (R)
  - **Symmetric** (sender and receiver know each other) vs. **Asymmetric** (only sender knows receiver)
  - Sender acts **synchronous** / **asynchronous**
- Types of Message Passing Models
  - Direct Addressing Model
  - Queue Communication Model
  - Port-oriented Communication Model

Middleware – What is it?

- Initial situation
  - Middleware germinated in the 1980s as a legacy system connection solution
  - Simplifies Distributed Processing, i.e. goal-oriented connection of numerous applications over a network
- Typical definitions
  - "Glue" between software components and the network
  - "/" (Slash) between Client/Server
  - Software platform bridging the heterogeneity of different systems and networks, which simultaneously provide a number of important system services, such as security policies, transaction mechanisms and directory services. [Schill & Spring]
- Typical forms
  - RPC Middleware
  - MOM (Message Oriented Middleware) through Message Queues
  - EAI (Enterprise Application Integration), for example: CRM,ERP, HR Adapter
  - Database Middleware
  - Middleware CORBA, JavaBeans, Enterprise JavaBeans, Microsoft COM

RPC-Middleware

- **RPC** – Programming language based approach that allows applications to **synchronal** call individual functions that are located in **separate processes** (not necessarily on the calling machine) using a **small channel** for exchanging input and output data.
- **IDL** – Interface Definition Language that expresses the function's signature, including input, output, and input/output parameters.
- **Semantic**
  - **Exactly-Once Execution and Exactly-Once Delivery**

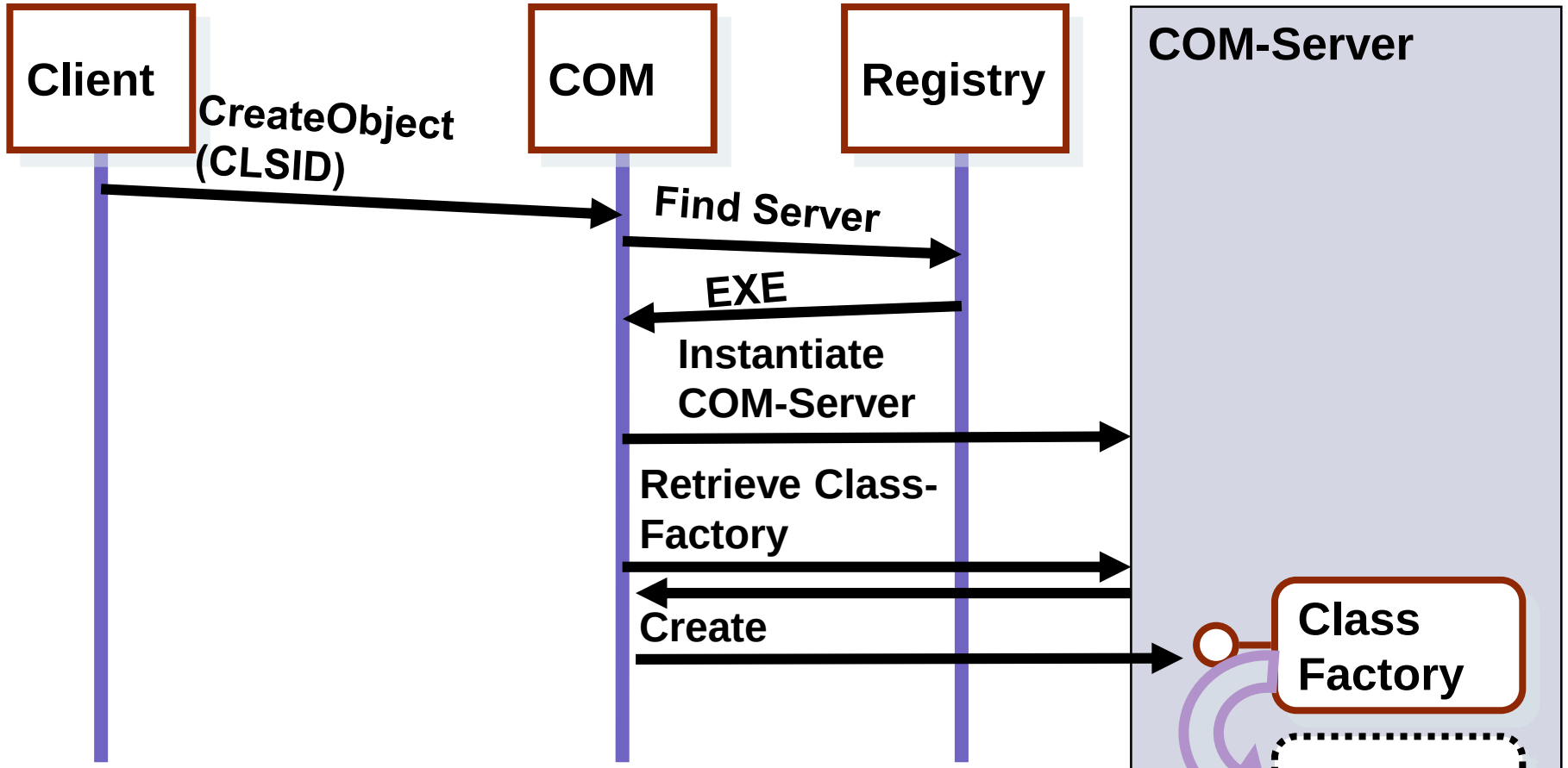


Component Middleware: COM

- Component Object Model (COM)
  - Microsoft's former standard Component-Technology
  - Developed out of OLE experience
- Components are referred to as so-called COM Server
  - The offered functionality is referred to as COM Class
  - COM Server contains one or more COM Classes
- Registry – Component database, saves components' location and meta data
  - Components must be registered in the registry
  - Components can be found through the registry

- COM Class
- Code implements COM interface
- Has a unique Id, so-called CLSID. Based on UUID
- An instance of COM Class is COM Object
- COM Servers
- Different types:
- In-Process Server (DLL is loaded in the Client Process)
- Out-of-Process Server (.Exe-File, is executed on the Client- or a remote machine (DCOM, COM+) with communication over RPC

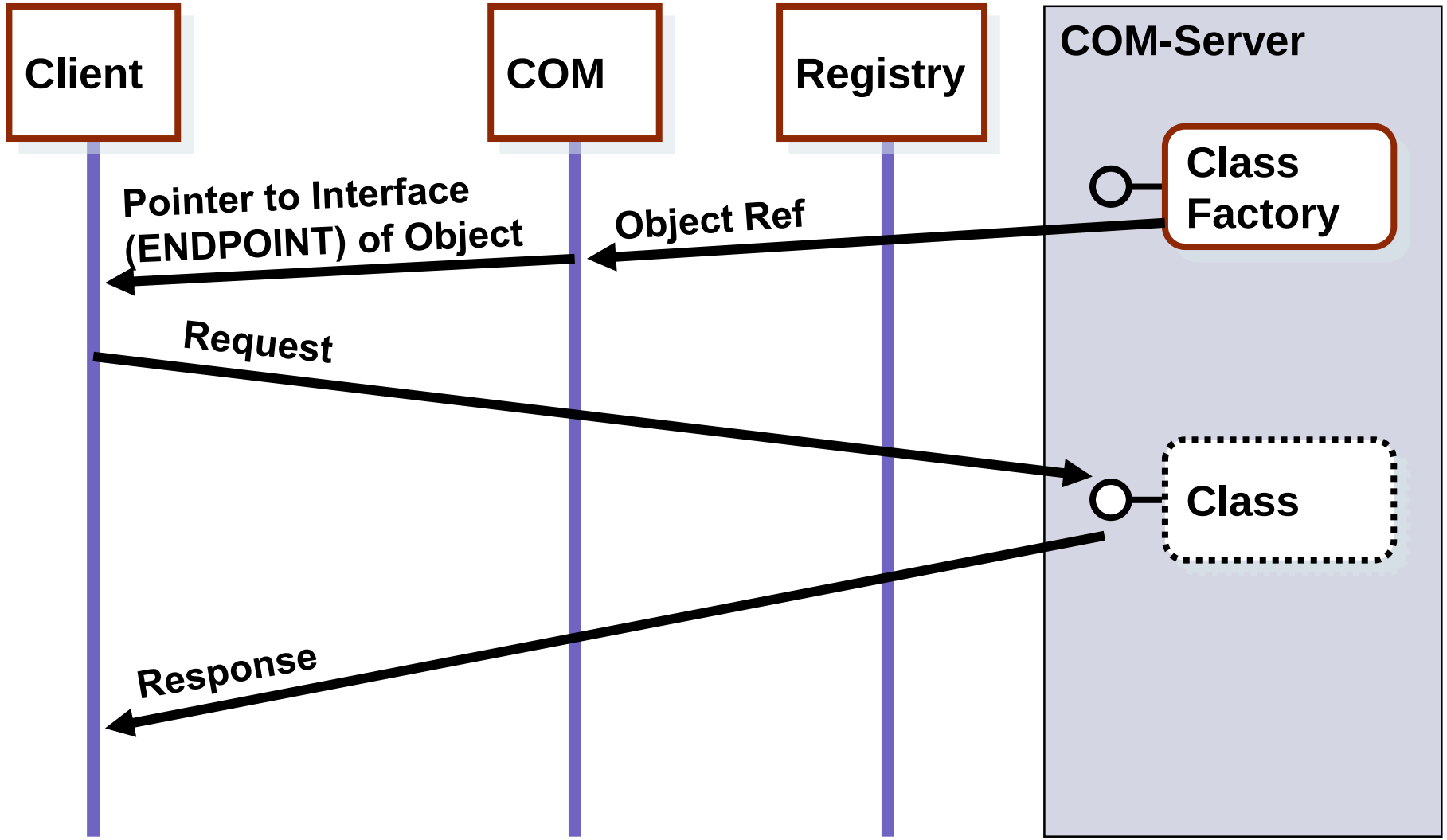
COM - Komponenten, Beispiel (1)



COM-Server (different forms possible):
- Local DLL in Client Process
- Remote DLL in Proxy Process

Warning: This is a simplified example meant to illustrate the use of components in COM!

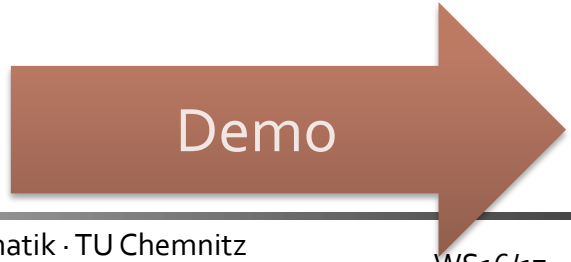
COM - Komponenten, Beispiel (2)



COM - Final Remarks

- Very important component technology
  - Runs on every Windows system (since NT,Win95)
  - Numerous improvements over standard RPC
  - Disadvantage: Proprietary, various additions: DCOM, COM+
- Endpoint in COM:
  - COM Interface defines the Contract
  - Contract consists of a set of methods and properties
  - Single component can provide multiple interfaces, but at least the IUnknown interface
  - IUnknown interface for component use/management: QueryInterface, AddRef, Release

- DEMO: Windows Registry



Other Component Middleware

- CORBA – Common Object Request Broker Architecture
  - Specification of OMG for interoperability between distributed computer systems
  - ORB: Middleware, which realizes the Requestor-Provider relation
  - CORBA – Problem: Inter-ORB incompatibilities call for restrictions
- Java Beans and Enterprise Java Beans
  - Based on the Virtual Machine principle
  - Application of principles similar to RMI
- Microsoft .Net Assemblies
  - Very interesting and powerful platform
  - Many modern concepts and a good class structure
  - Other platforms partially adopt the approach (such as Mono)

Next:

- How to call remote functions using the web
- Why there is more than calling functions

PART II

SSE-Technology-Basics

Chapter 1

WHAT IS SOA AND SERVICE FROM THE TECHNOLOGY POINT OF VIEW

Intro.: SOA in the context of the Web

- **Service Oriented Architecture (SOA)** – Architectural concept (typically business-driven), which concentrates on systematic service use and provisioning by SOA's participants
  - Concept is independent of the technology in use – only the relations between the participating Service Provider, Service Broker and Service Consumer are being considered
  - SOA comes in many flavors, for example, SOA with components (see Microsoft DCOM/COM+)

What is SOA

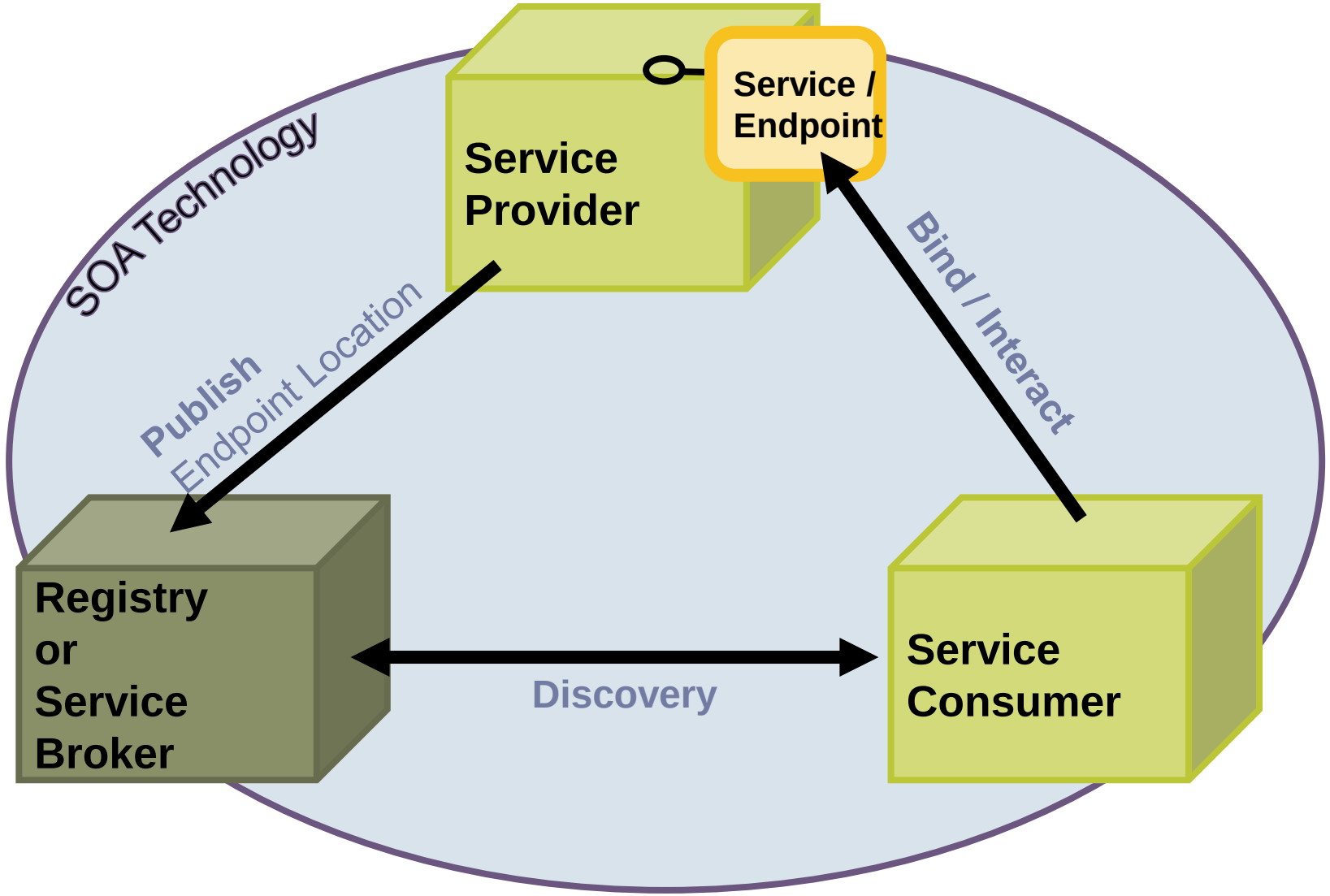
From the point of view of:	SOA is
Business executive and business analyst	A set of services that constitutes IT assets (capabilities) and can be used for building solutions and exposing them to customers and partners
Enterprise architect	A set of architectural principles and patterns addressing overall characteristics of solutions: modularity, encapsulation, loose coupling, separation of concerns, reuse, composability, and so on
Project manager	A development approach supporting massive parallel development
Tester or quality assurance engineer	A way to modularize, and consequently simplify, overall system testing
Software developer	A programming model complete with standards, tools, and technologies, such as Web services

Source: Lublinsky, Boris. "Defining SOA as an Architectural Style." IBM developerWorks, January 2007.

Introduction: SOA in Web-Context

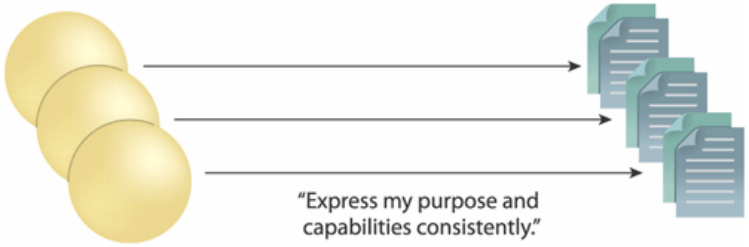
- **SOA Service** – Autonomous, self-contained, reusable software system (so-called "Black-box"), which enables (business-)task support, whereby functions for use by Service Consumer are provided by means of specified message-exchange methods
  - Service implementation must meet the requirements of the SOA environment
  - Service access (typically) over a network
  - Service description is available, for example, via a registry
- SOA service in Web-context: **Web Service**

SOA - Overview

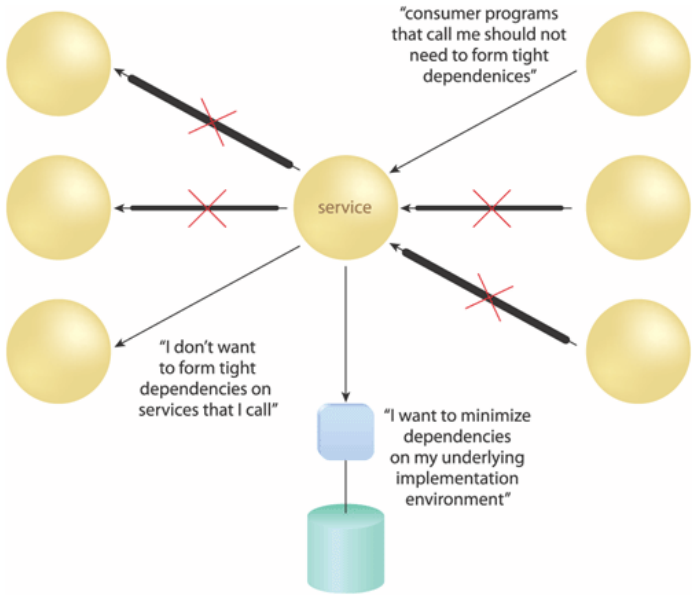


SOA Principles

- Standardized Service Contracts
  - Services should expose contracts describing their purpose and capabilities and which comply with organization-wide schemas and policies
    Goals: consistency, reliability and efficient governance



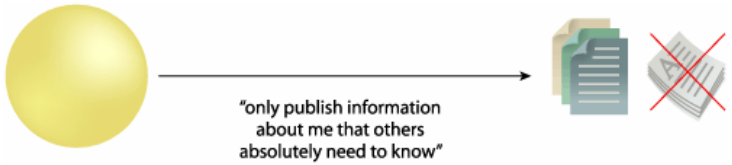
- Loose coupling
  - Service contracts should avoid dependencies on service consumers and underlying implementation environments
    Goals: service interoperability and reliability



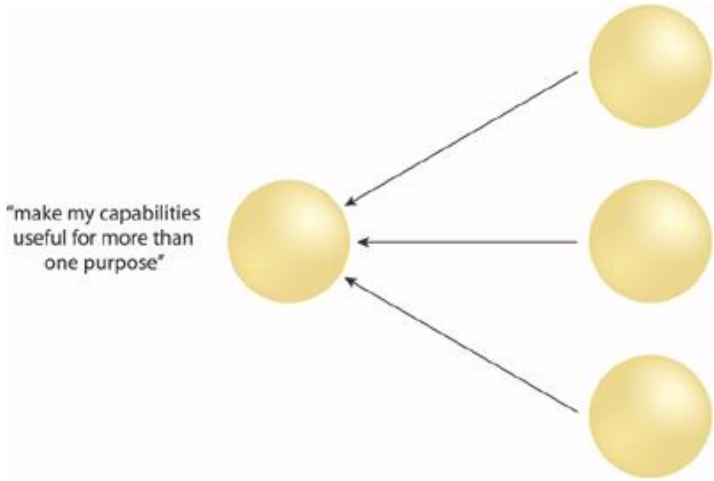
Source: SOA Principles of Service Design by Thomas Erl

SOA Principles (2)

- Abstraction
  - Service contracts should expose only minimally required information for its consumption and hide unnecessary details
    Goals: enable loose coupling and efficient service composition



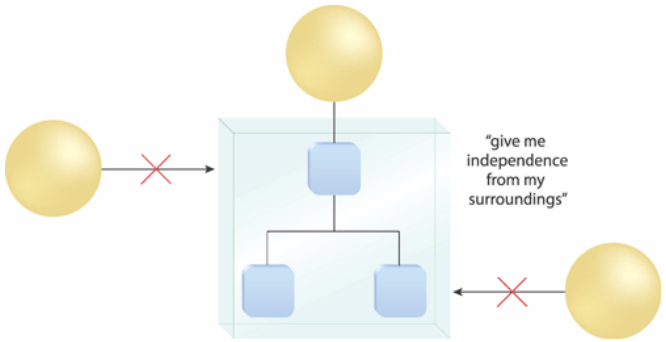
- Reusability:
  - Services should implement generic and context-agnostic logic
    Goals: costs- and time-saving, efficient service composition



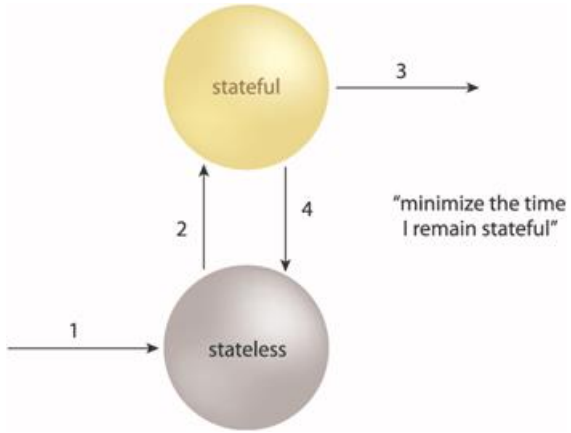
Source: SOA Principles of Service Design by Thomas Erl

SOA Principles (2)

- Autonomy
  - Services have a high control of their underlying execution environment
    Goals: reliability, predictability



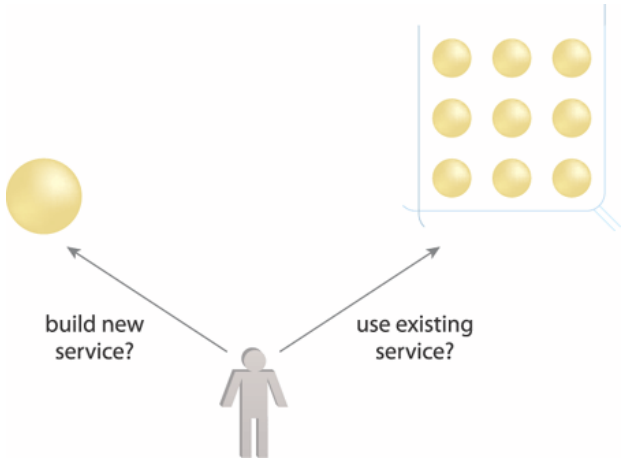
- Statelessness
  - Services minimize resource consumption by deferring the management of state
    Goals: performance, scalability



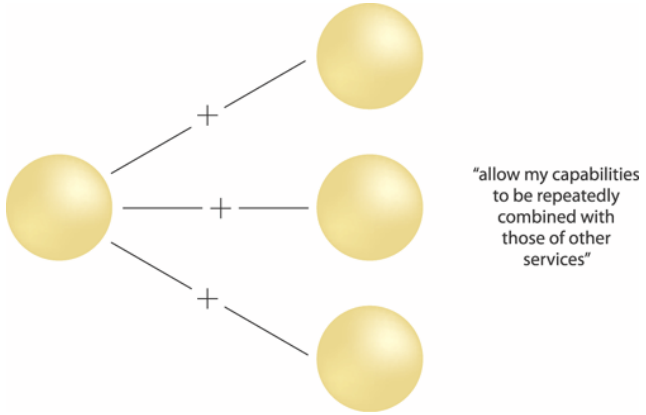
Source: SOA Principles of Service Design by Thomas Erl

SOA Principles (2)

- Discoverability
  - Services provide both human and machine-readable metadata, which can be used by discovery agents
    Goals: reusability, efficient service composition



- Composability
  - Services contracts and implementation are designed with the goal to be composed with others
    Goals: reusability, loose coupling



Source: SOA Principles of Service Design by Thomas Erl