



Universität Münster
Institut für Wirtschaftsinformatik

Lehrstuhl für Wirtschaftsinformatik
und Interorganisationssysteme
Prof. Dr. Stefan Klein

www.wi-ios.de
mail@wi-ios.de

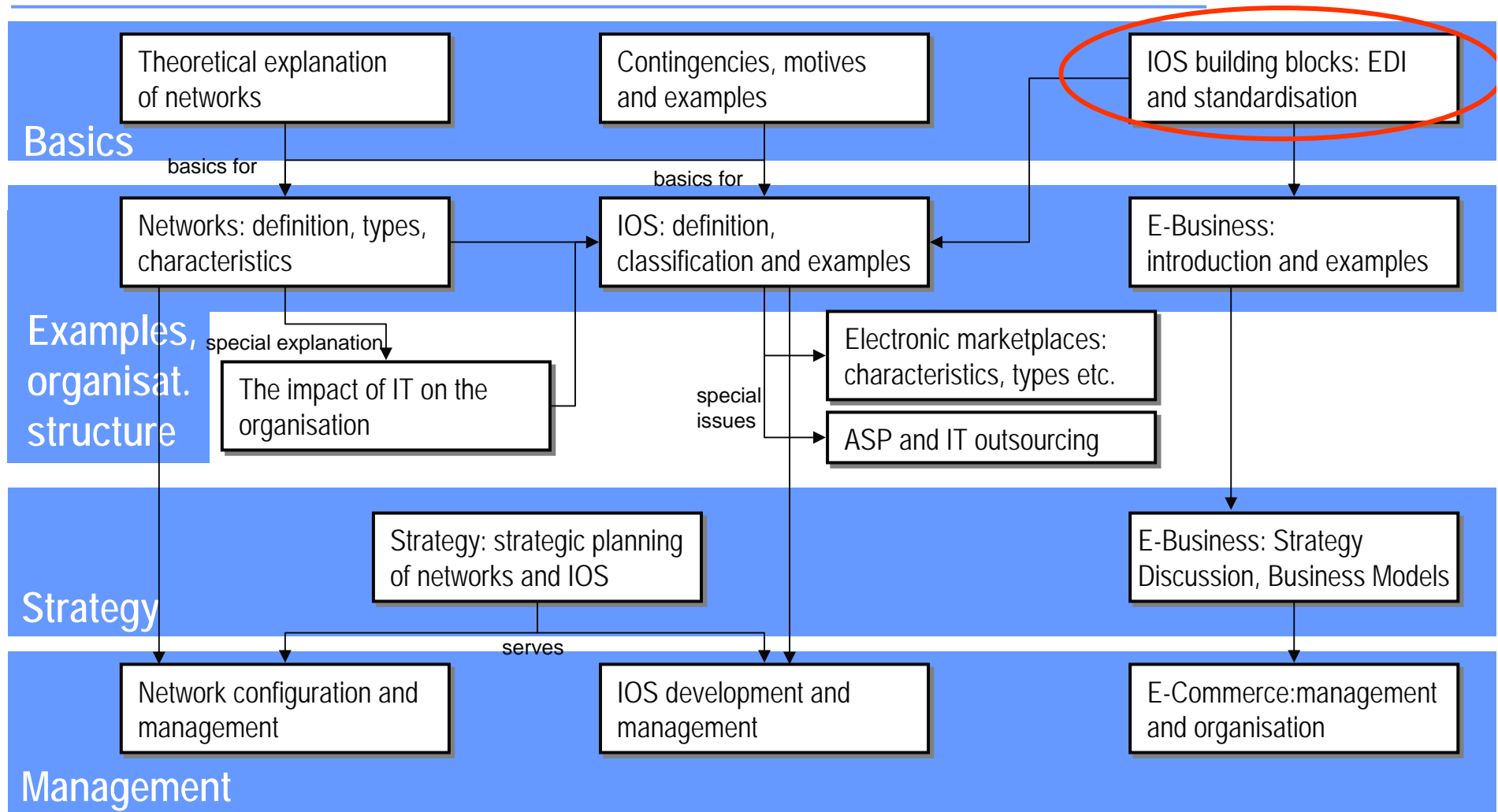


Building Blocks of Inter-Organisational Systems Part 2

Dr. Hans-Dieter Zimmermann
Lehrstuhl für Wirtschaftsinformatik und
Interorganisationssysteme
Institut für Wirtschaftsinformatik
Universität Münster



Course Outline



Objective

- What are (IT) building blocks for IOS?
- Visions and scenarios that have guided the development of IOS technologies.
- The role of standards.
- Generations and development trends of IOS technologies.

Agenda

A. Electronic Data Interchange (EDI)

B. Enterprise Application Integration (EAI)

C. Middleware

Agenda

A. Electronic Data Interchange (EDI)

B. Enterprise Application Integration (EAI)

C. Middleware

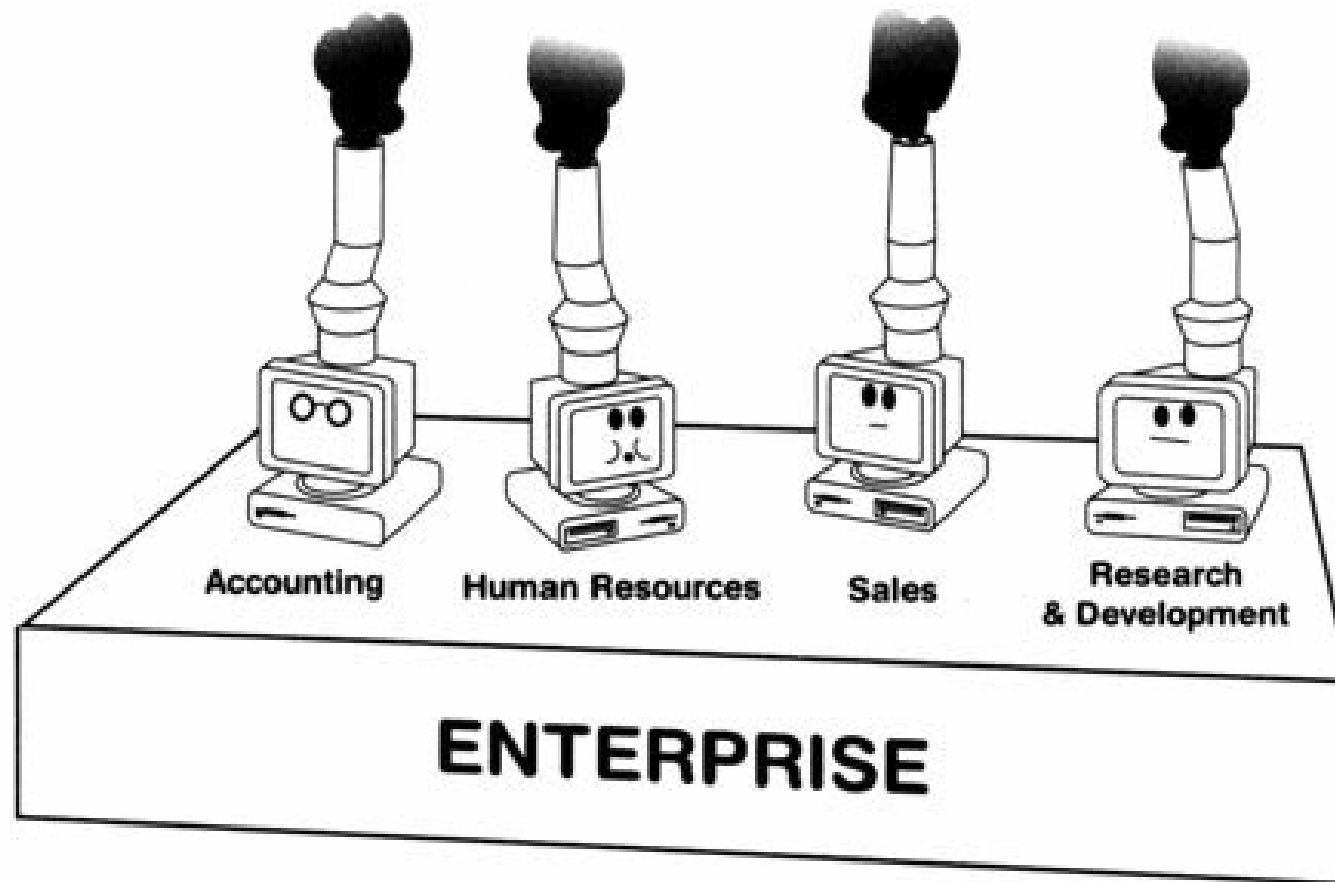
Inter-application communication

Current situation: An information system is mainly composed of point-to-point communication between applications

Historical reasons

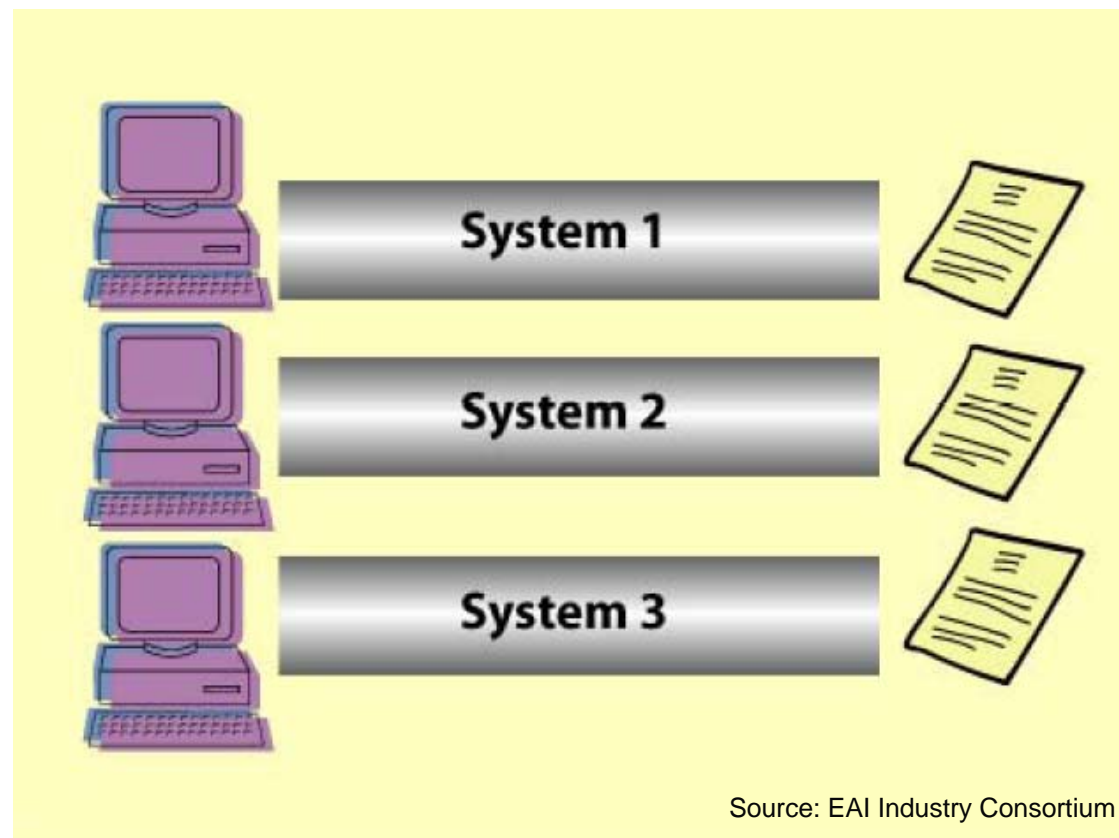
- Evolution of “stovepipe” applications (different business functions have their own applications, e.g. accounting, sales, human resources)
- Need for communication between these applications
- Establishing of peer-to-peer communication; usually hand-coded in COBOL, C, etc.
- Resulting in “spaghetti integration”

Stovepipe applications



Stovepipe applications

- Each system has its own data store and produces paper documents that are shared between people to perform the business function.



Stovepipe applications

- Typically custom-specific/”departmentalised” solutions
- Often proprietary
- Utilising the technology-of-the-day (mainframes, UNIX servers, NT servers, etc.)
- Using nonstandard data storage and application development technology
- “Software Monoliths”

Examples

- Inventory control
- Sales automation
- General ledger
- Packaged applications such as ERP applications (e.g. SAP): information sharing often is limited to their own proprietary technology

Technological development

Early days of computing

- Information processing on centralised platforms (mainframes)
- Processes and data exist in a homogenous environment
- Integrating applications quite simple (some additional coding)
- Long-term strategy

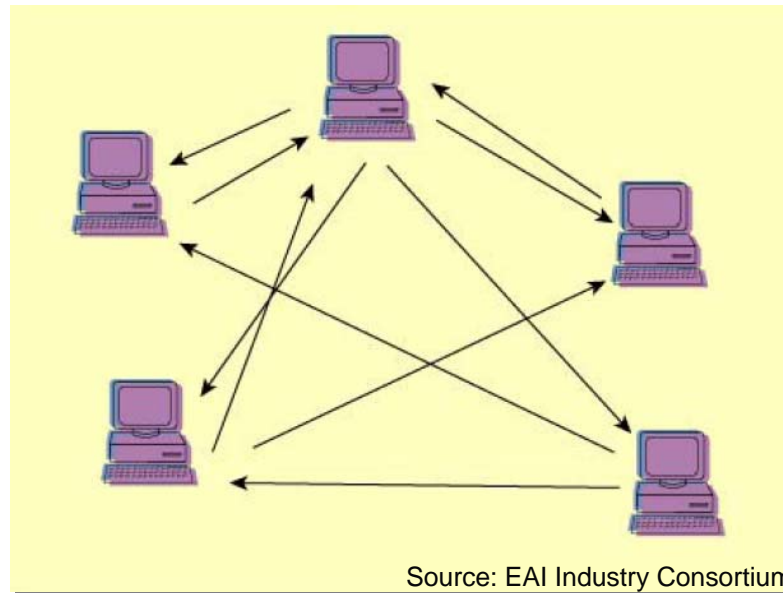
Nowadays

- Smaller and more open platforms (UNIX, Windows NT)
- New programming paradigms (OOP, component-based development)
- Need for integration with existing, older systems
- Often minimal architectural foresight to the selection of platforms and applications (implementing most popular technologies)

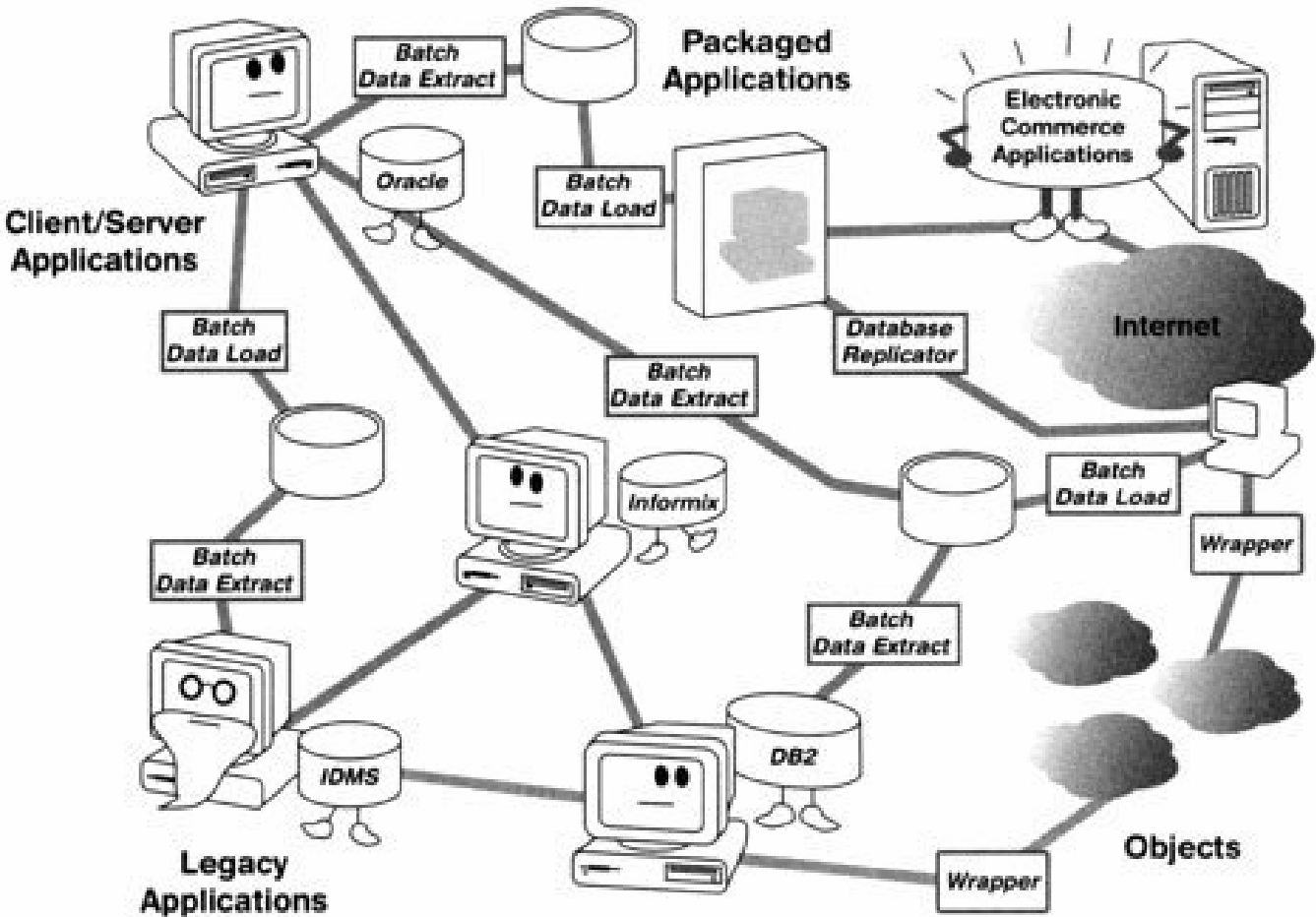
Spaghetti integration

End result

- Enterprise IS are a mixing of technologies and paradigms
- “Point-to-point” solutions to create single links between many applications
- Maintaining the linkages becomes more expensive than maintaining the linked applications

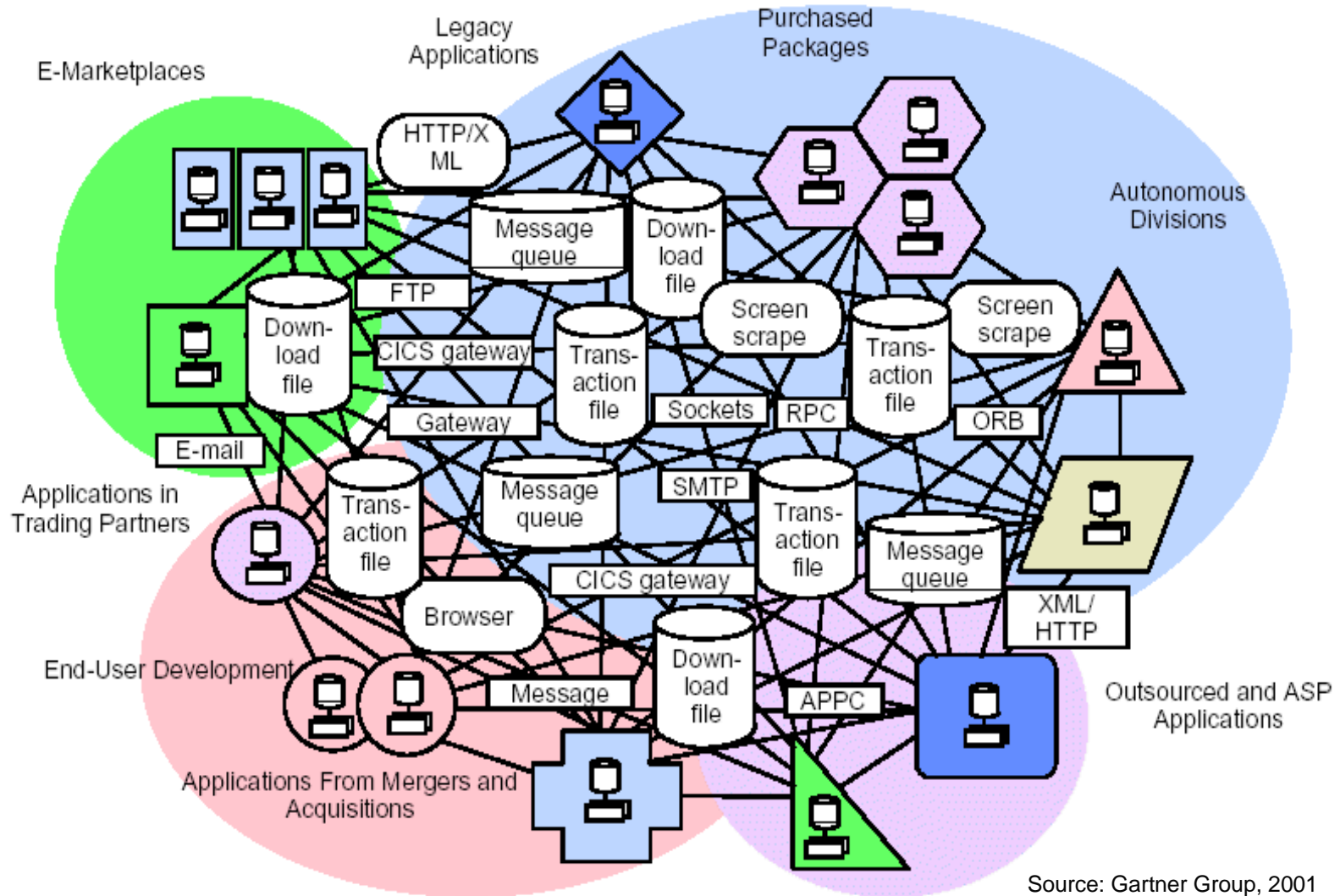


Spaghetti integration



Linthicum (1999), p. 9

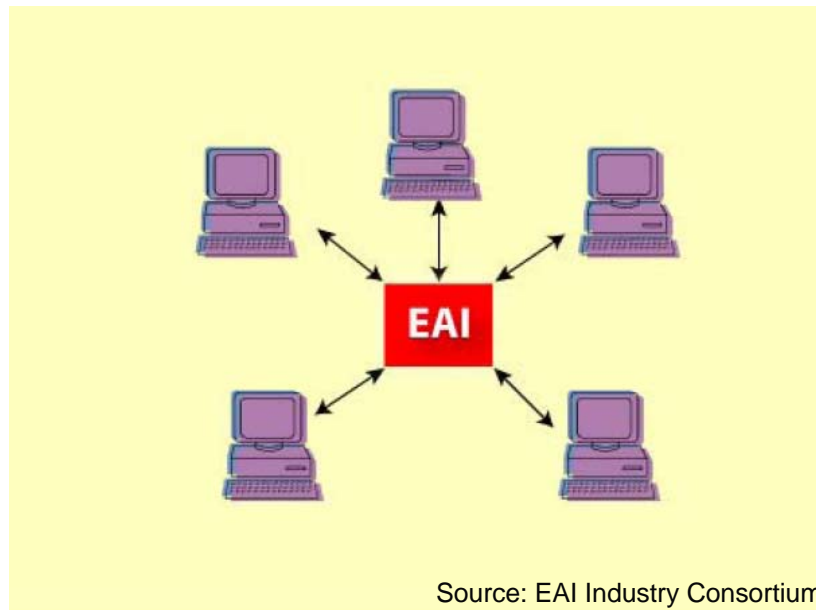
Spaghetti integration



What is EAI?

- “EAI is an approach and technology that is used to link disparate enterprise systems into a single cohesive infrastructure focused on fulfilling business needs.”
- “Enterprise Application Integration (EAI) standardizes and simplifies integrating multiple systems by using a software system that coordinated all interactions.”

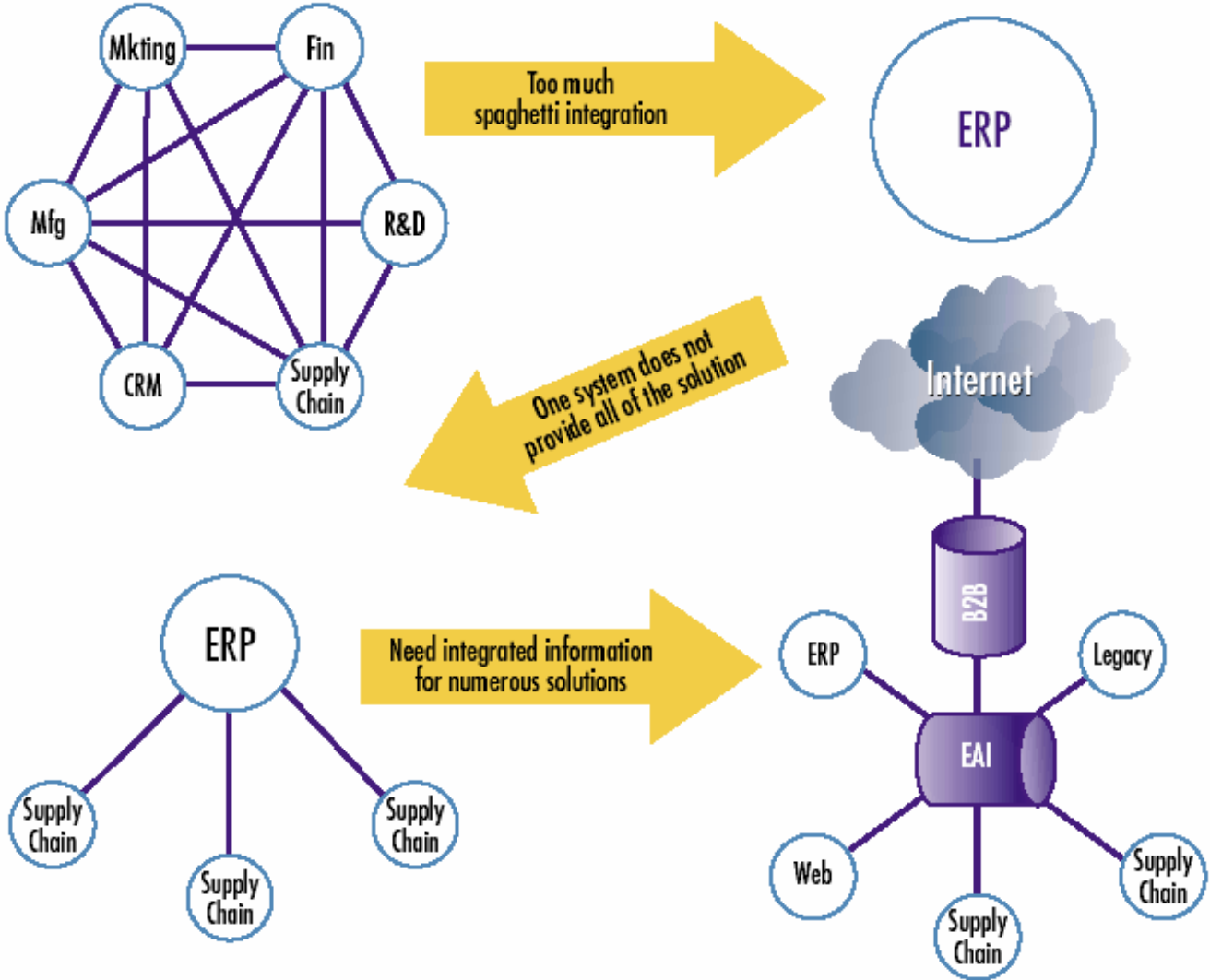
[Source:
EAI Industry Consortium]



What is EAI?

- EAI Solutions leverage a framework-based approach to accomplish:
 - Straight through processing, which allows the propagation of real-time information throughout the enterprise
 - The capability to extend the enterprise to have a closer relationship with employees, customers and partners
 - Deploying tools to lower the total cost per transaction, resulting in higher returns on technology investments
 - The flexibility to incorporate new leading edge technologies, while minimizing risk of failed user acceptance

The way to EAI



Pinkston (2001), p. 49

The way to EAI

1. Departmental applications

- Lead to “spaghetti integration”

2. ERP systems

- Claimed to be a one-system-fits-all solution
- But ERP systems do not have all the functionalities needed to support the business processes
- ERP has to be integrated with legacy or new systems

3. Middleware/ EAI

- Providing an infrastructure to connect and interface information between an organisation’s internal applications
- Need for B2B connectivity: integrating applications and business processes with business partners

Enterprise Application Integration

Aim: Integrate existing - both intra- and inter-organisational - applications within an enterprise using a common middleware rather than recreate the same business processes and data repositories over and over.

Reasons

- saving development costs
- existing value of legacy applications - albeit using “ancient” technologies
- increasing need for integration by the popularity of packaged applications such as SAP
- need for a comprehensive integration system rather than creating interfaces and points of integration between every application and data source

Ultimate EAI scenario: a common virtual system

Enterprise Application Integration

Basic requirements for EAI

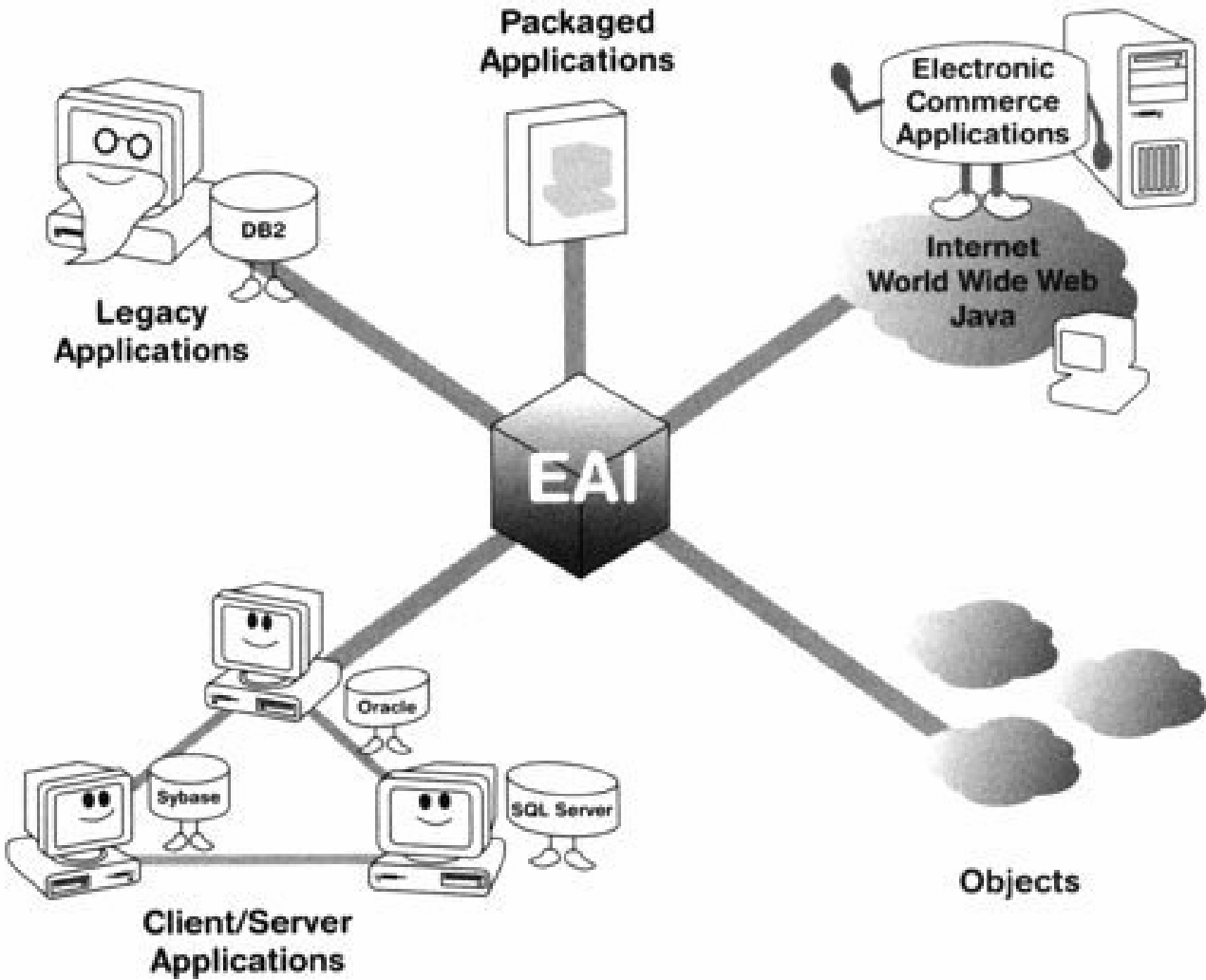
1. Understanding large-picture architecture

- Understanding processes and data that exist in the enterprise
- Determining which applications and data stores need to share information
- But interdepartmental turf battles, changing business requirements etc. make EAI difficult

2. New technology to solve EAI problem

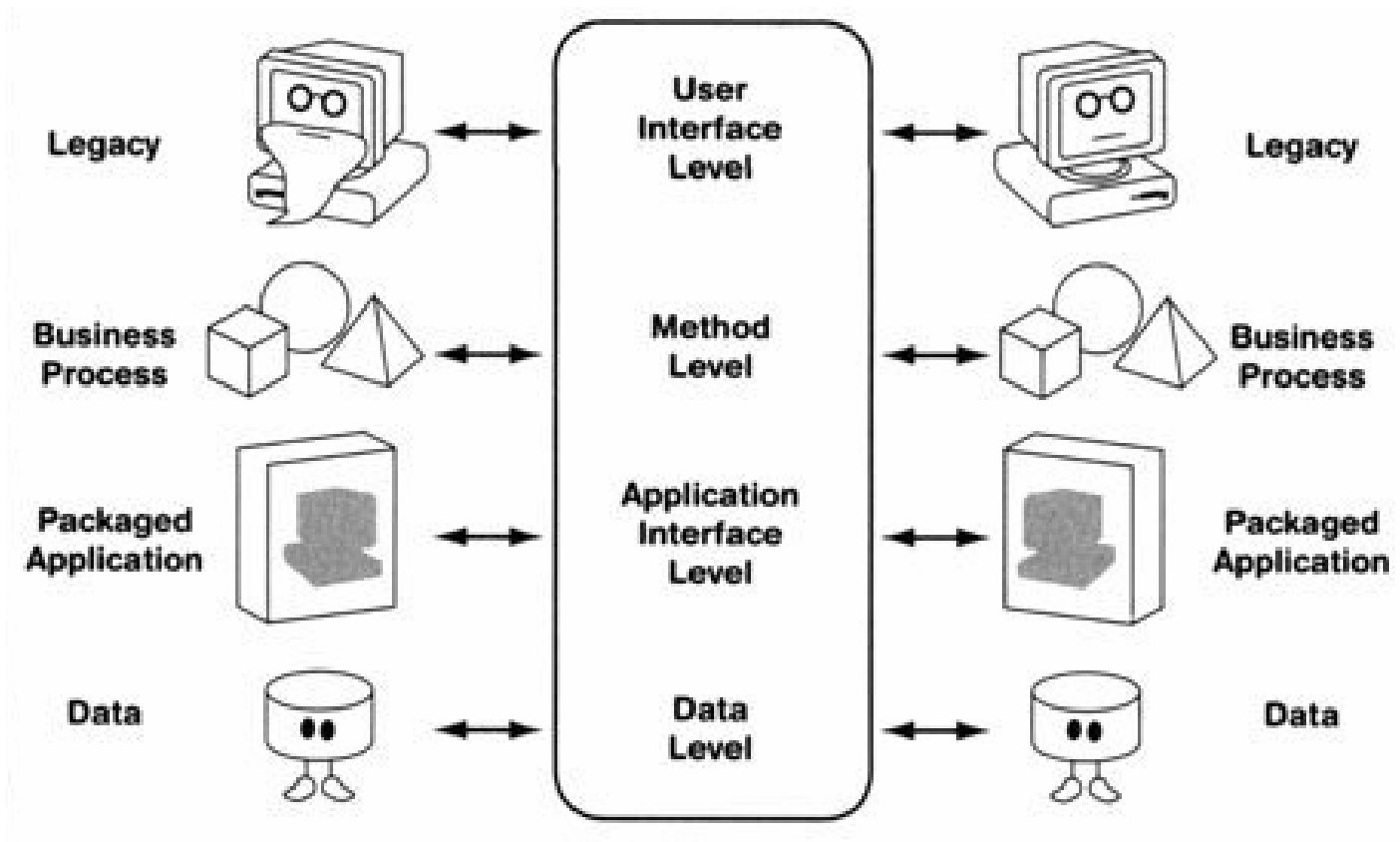
- Message brokers move messages from one system to another and convert their message formats
- New generation of application servers and distributed object technologies enables process integration

Message broker



Linthicum (1999), p. 10

Types of EAI



Linthicum (1999), p. 19

Types of EAI

Data-level EAI

- Moving data between data stores
- Extracting data from one database, processing data as needed, updating data in another database
- Relatively inexpensive, no necessity of changing application code

Application interface-level EAI

- Accessing business processes and simple information
- Bundling many applications together
- Using interfaces (e.g. APIs) of custom or packaged applications to access processes and data, extract the information, and - after a format conversion - transmit them to the target system
- Specific features and functions of application interfaces limit integration

Types of EAI

Method-level EAI

- Sharing of business logic
- For example a method for updating a customer record may be accessed from a number of applications
- Mechanisms: distributed objects, application servers, transaction processing monitors, etc.
- Challenging technologic issues

User interface-level EAI

- Bundling applications by using their user interfaces (screen scraping)
- For example a mainframe application that do not provide database- or business process-level accessed may be accessed by the user interface
- No knowledge about application design etc. needed
- Primitive, might be an unstable and archaic approach

Agenda

A. Electronic Data Interchange (EDI)

B. Enterprise Application Integration (EAI)

C. Middleware

Middleware: Introduction

Middleware can be defined as a layer of software whose purpose is to mask heterogeneity and to provide a convenient programming model to application programmers.

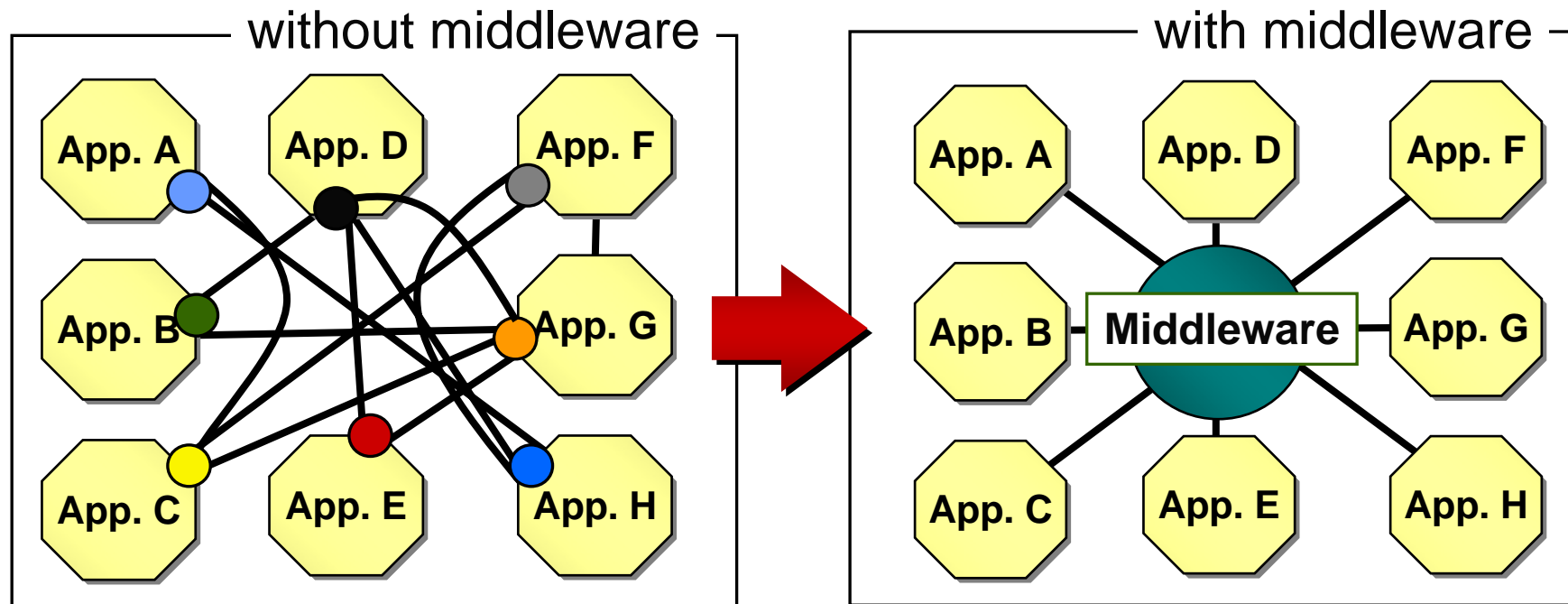
(cf. Coulouris et al. 2000)

Classification of middleware concepts

- **Database middleware** performs only database access
- **Transaction processing (TP) monitors** middleware supports queues that enforce message delivery
- **Message-oriented middleware (MOM)** is an event-driven, asynchronous, non-blocking and message-based communication method that guarantees message delivery
- **Remote procedure call (RPC)** is a method that allows a machine to request service from another machine
- **Object request broker (ORB)** is a tool that passes requests from clients to the object implementations on which they are invoked

Middleware: The answer to complexity

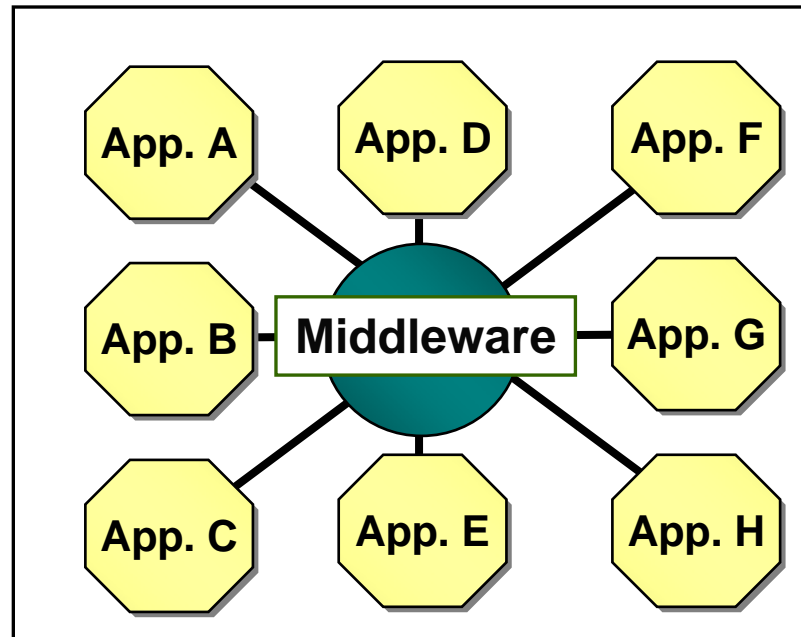
- Integration Middleware addresses the complexity of each application handling it's own communication and data mapping



All issues involved with applications communicating are handled in a centralized 'hub' (middleware)

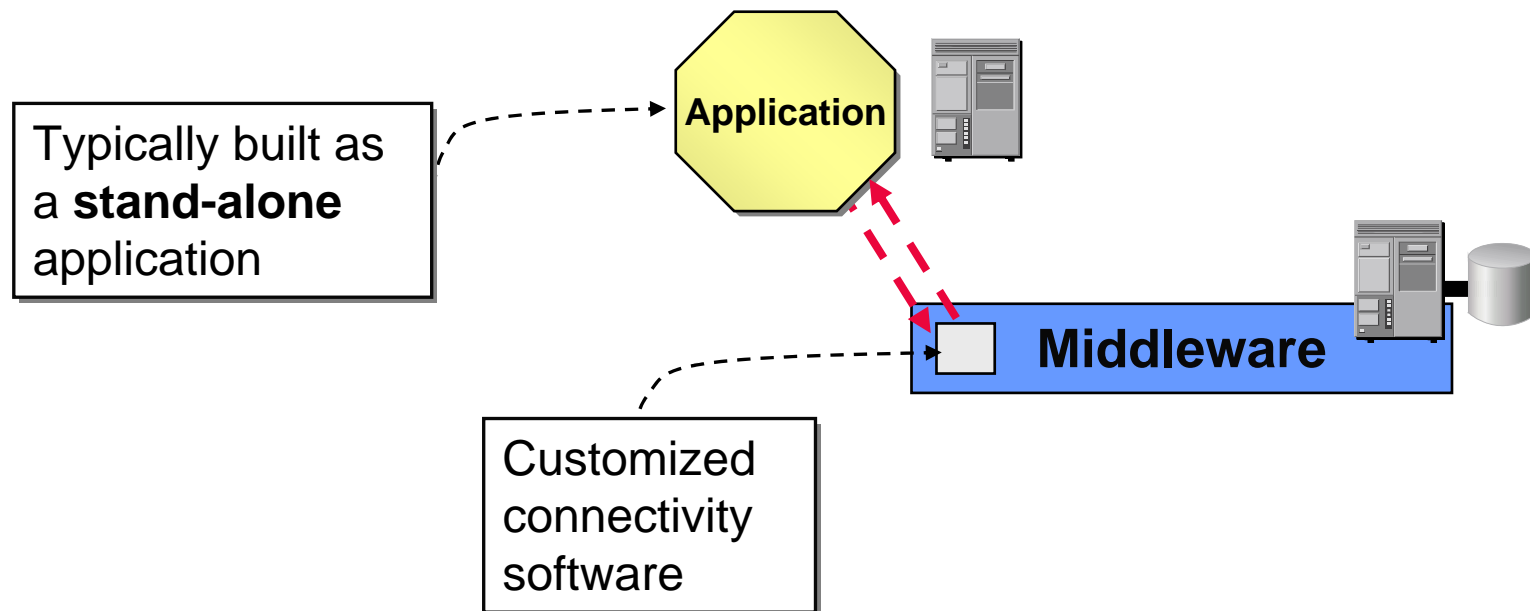
Middleware

- Integration Middleware:
An application that connects with and coordinates the work of the other application systems



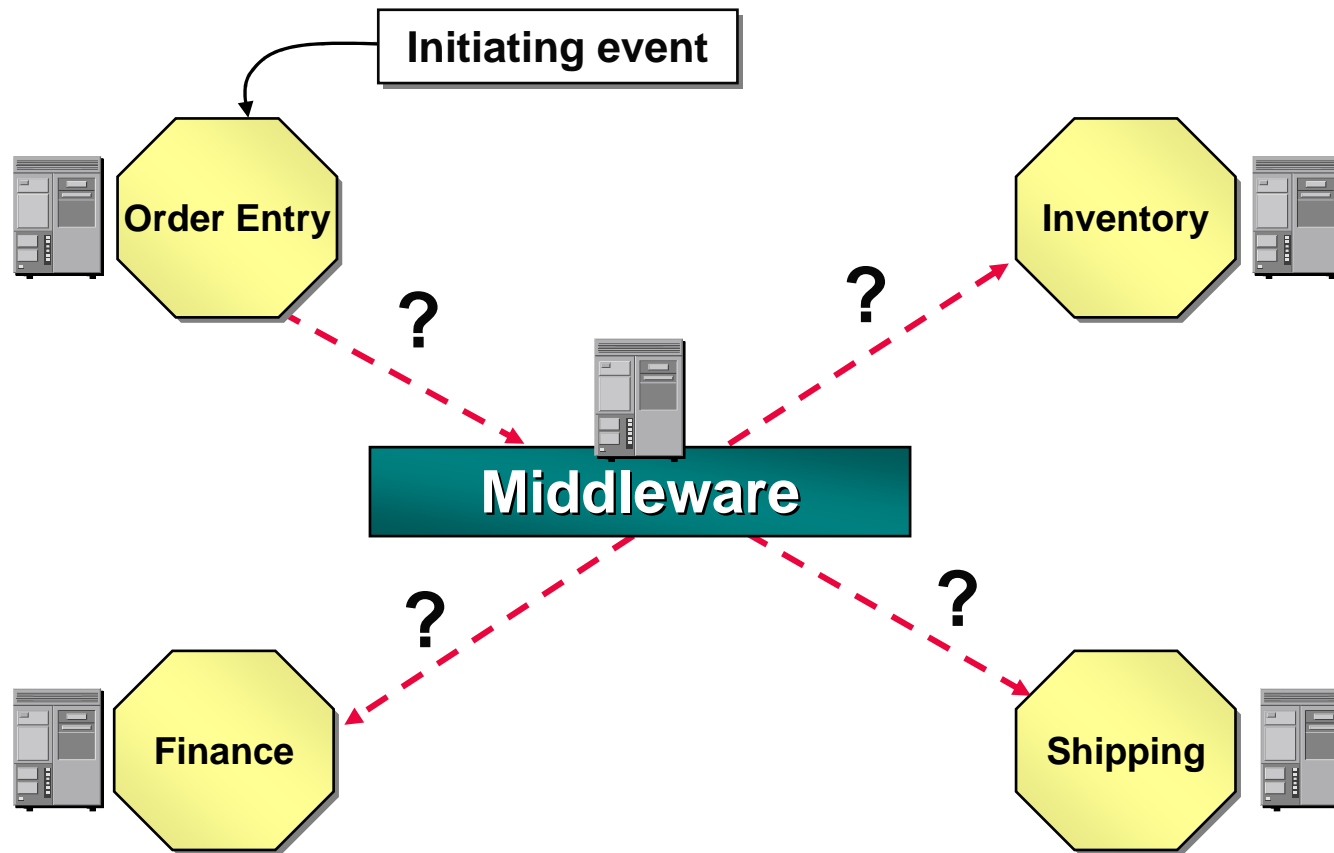
Middleware Connects Applications

- Middleware provides connectivity and secure and reliable transport of data (encryption, re-send of data, storage of data)



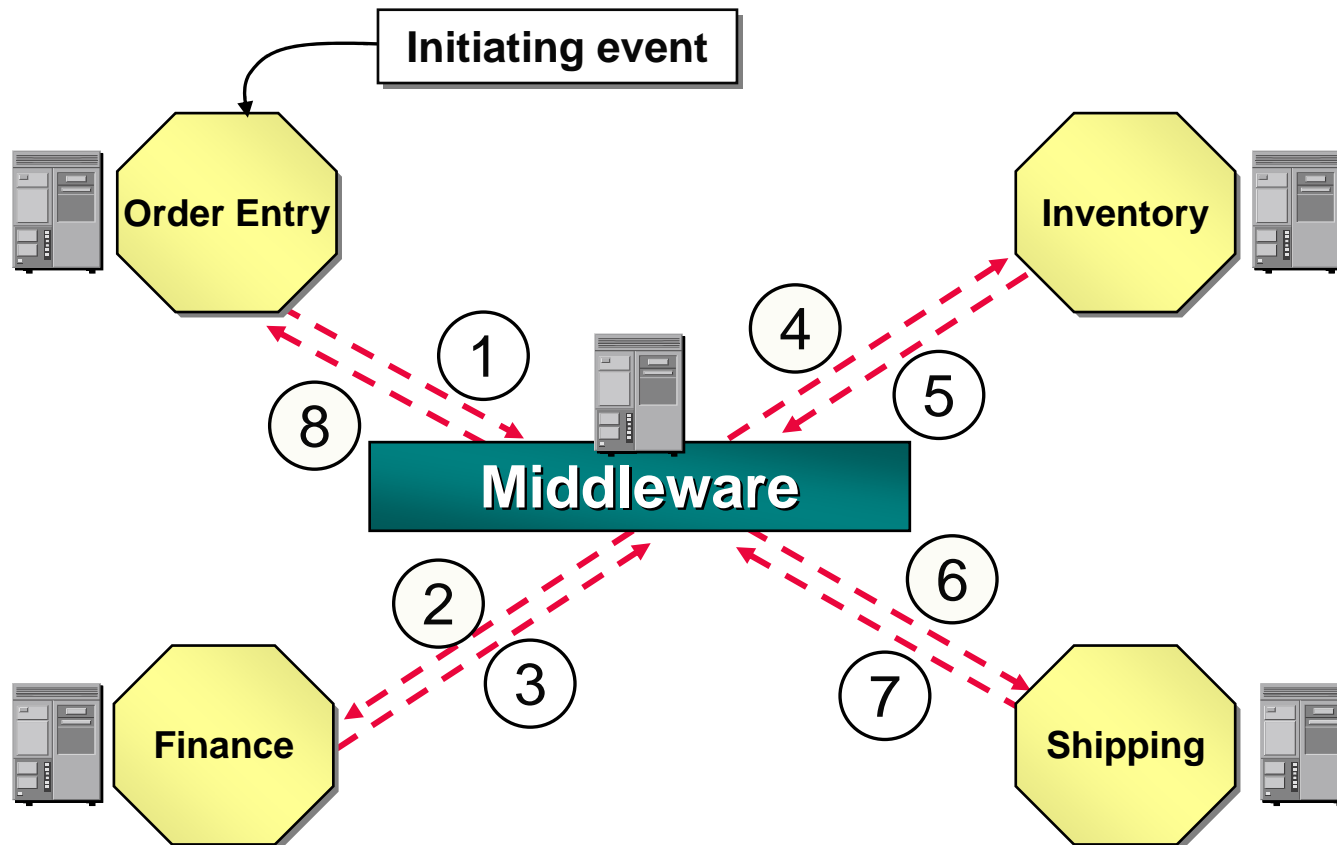
Middleware Coordinates Applications

- What happens, to which applications, and in what order?



Middleware Coordinates Applications

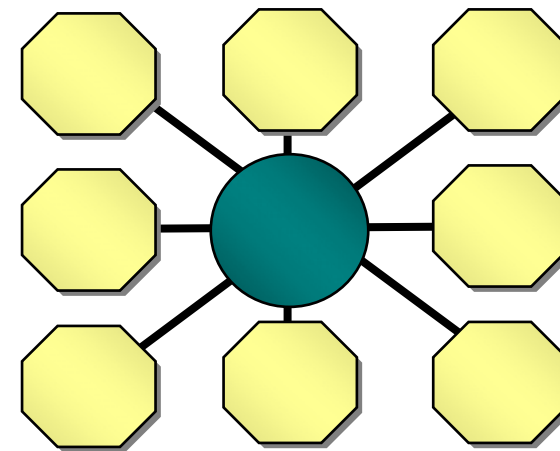
- Example steps in achieving the goal: Process an order



EAI Middleware Requirements Summary

What functions must integration middleware provide?

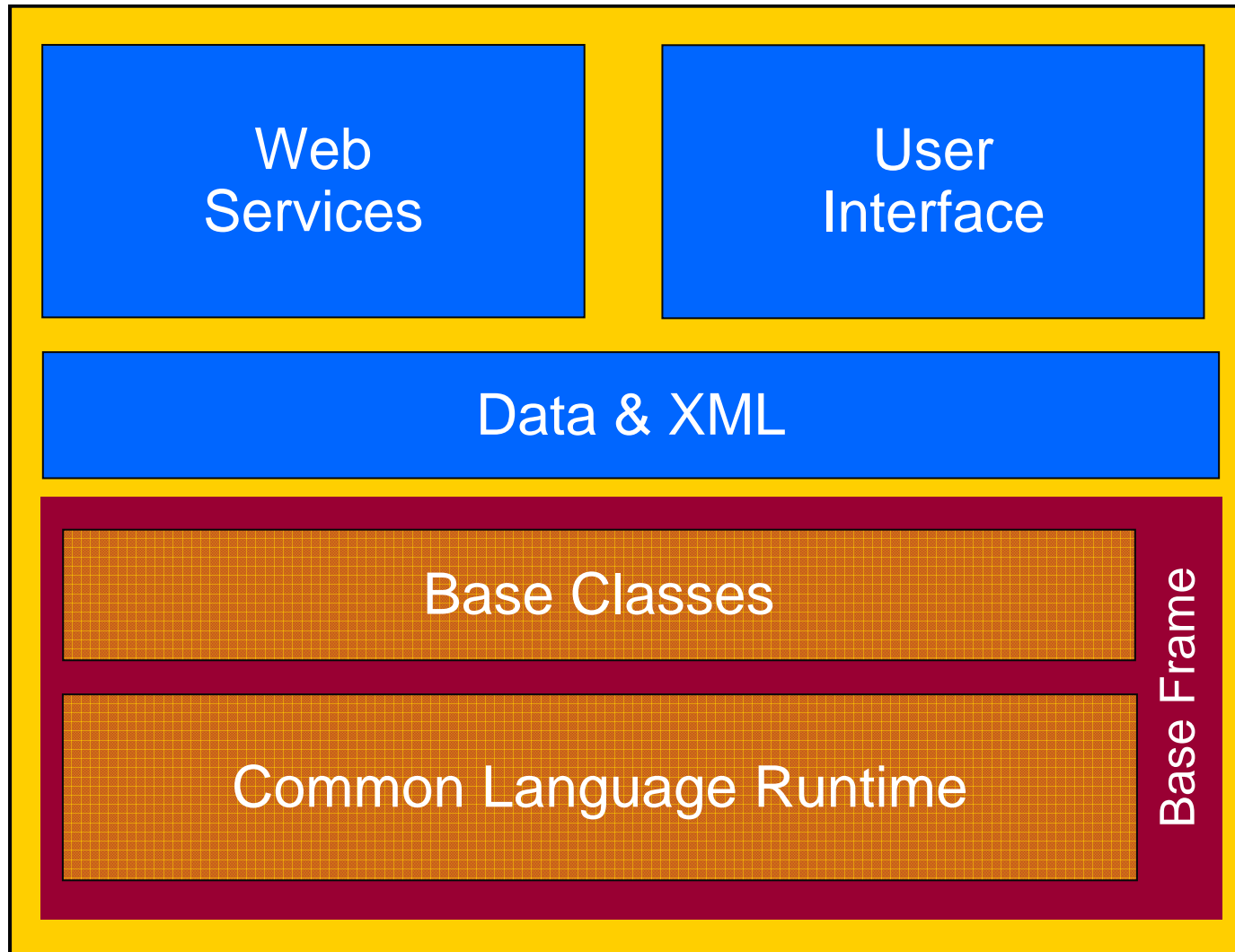
- Allow applications to connect to the middleware hub
- Transform/ map data
- Enable modeling of business goals (“Business Process Modeling (BPM)”) and routing capabilities to enable messages to be sent to the correct target application.
- Provide transactional support
- Error detection and correction



The Microsoft .NET framework: What Is .NET?

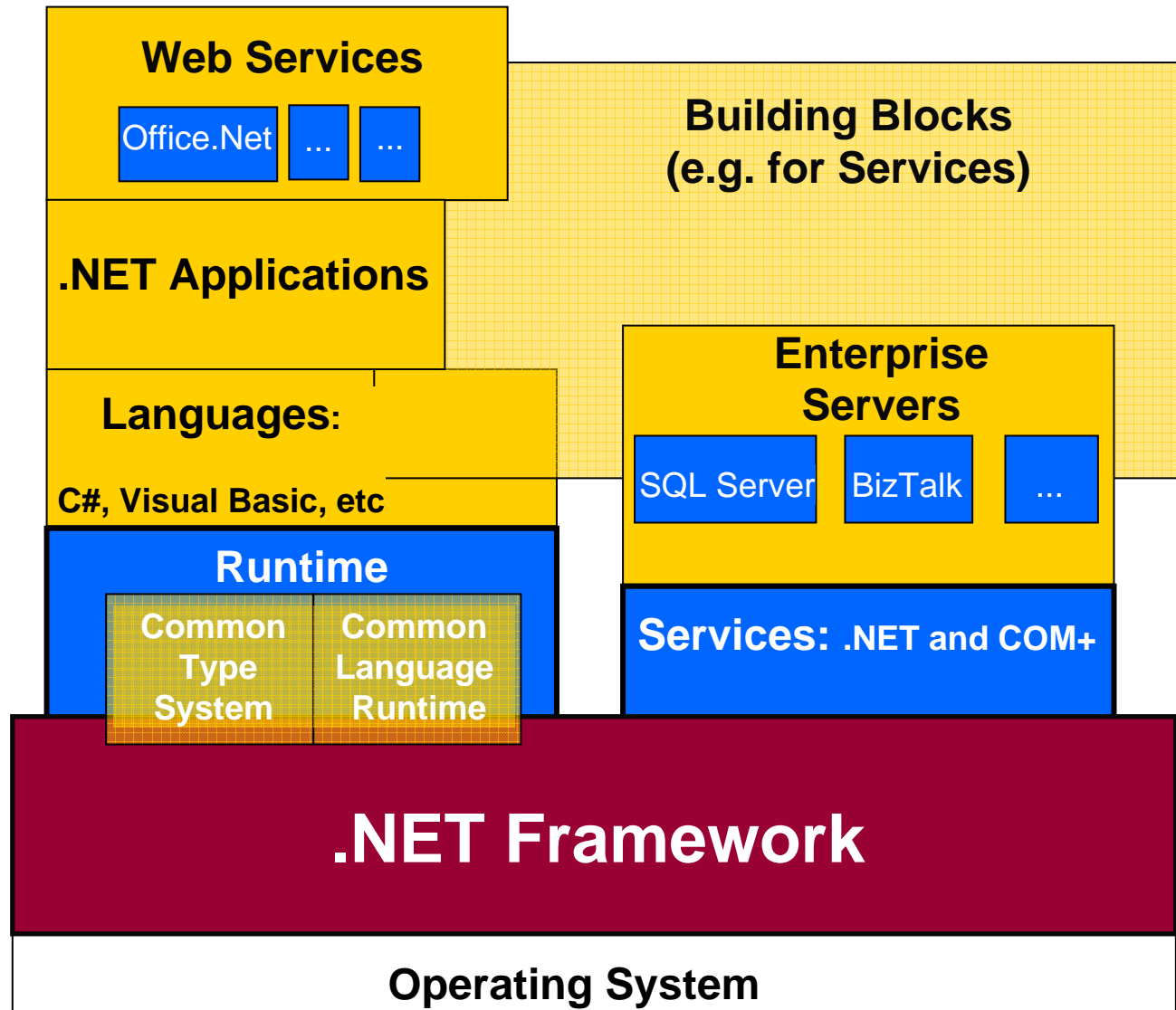
- The combination of:
 - Framework
 - Common language runtime
 - Class libraries
 - ASP.NET
 - Web Services
 - .NET Enterprise Servers
- “The means to build the Web the way you want it!”

Microsoft .NET Framework Diagram



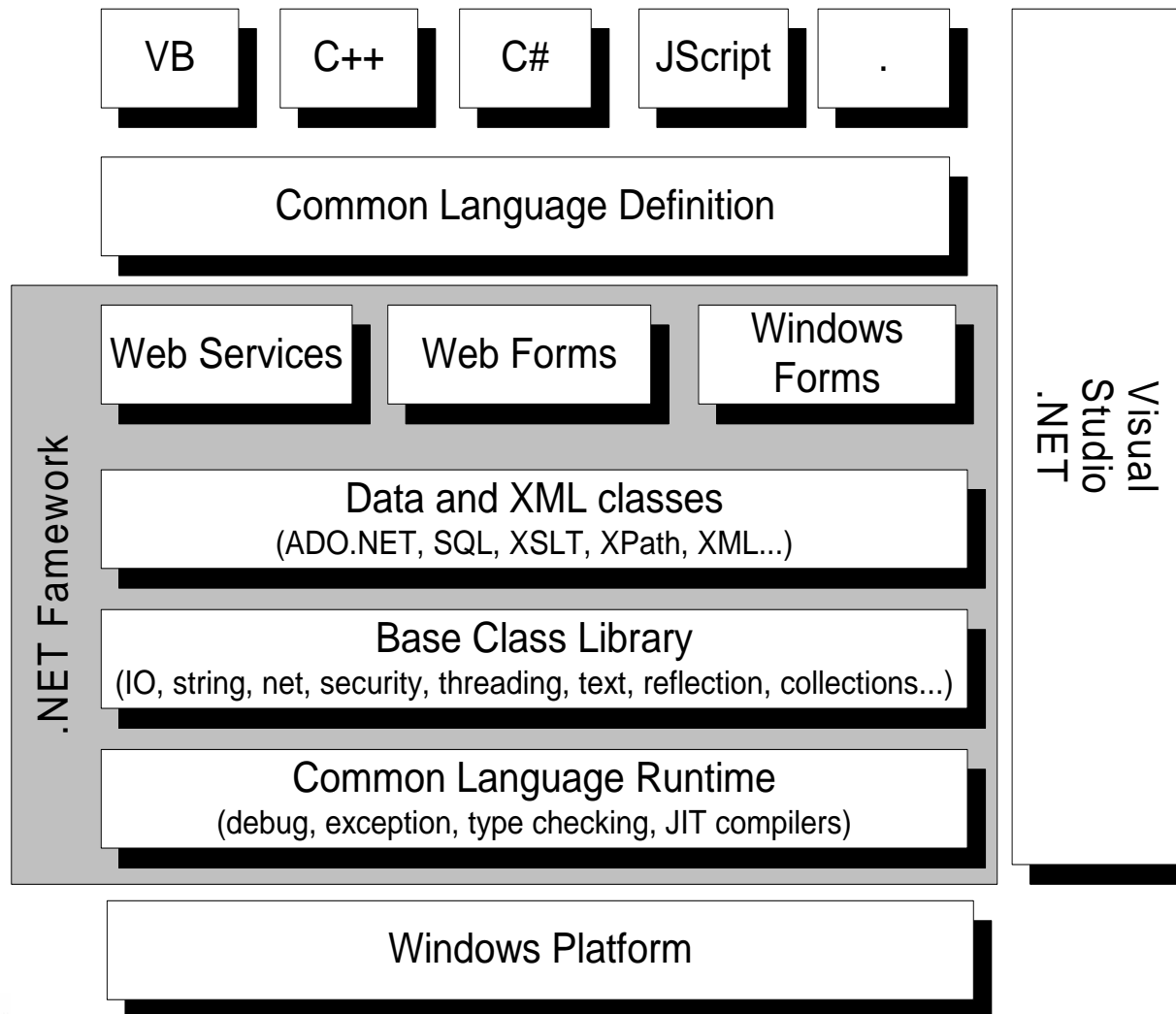
Source: Microsoft

Microsoft .NET and the .NET Framework



Source: Microsoft

Microsoft .NET framework



Microsoft .NET summary

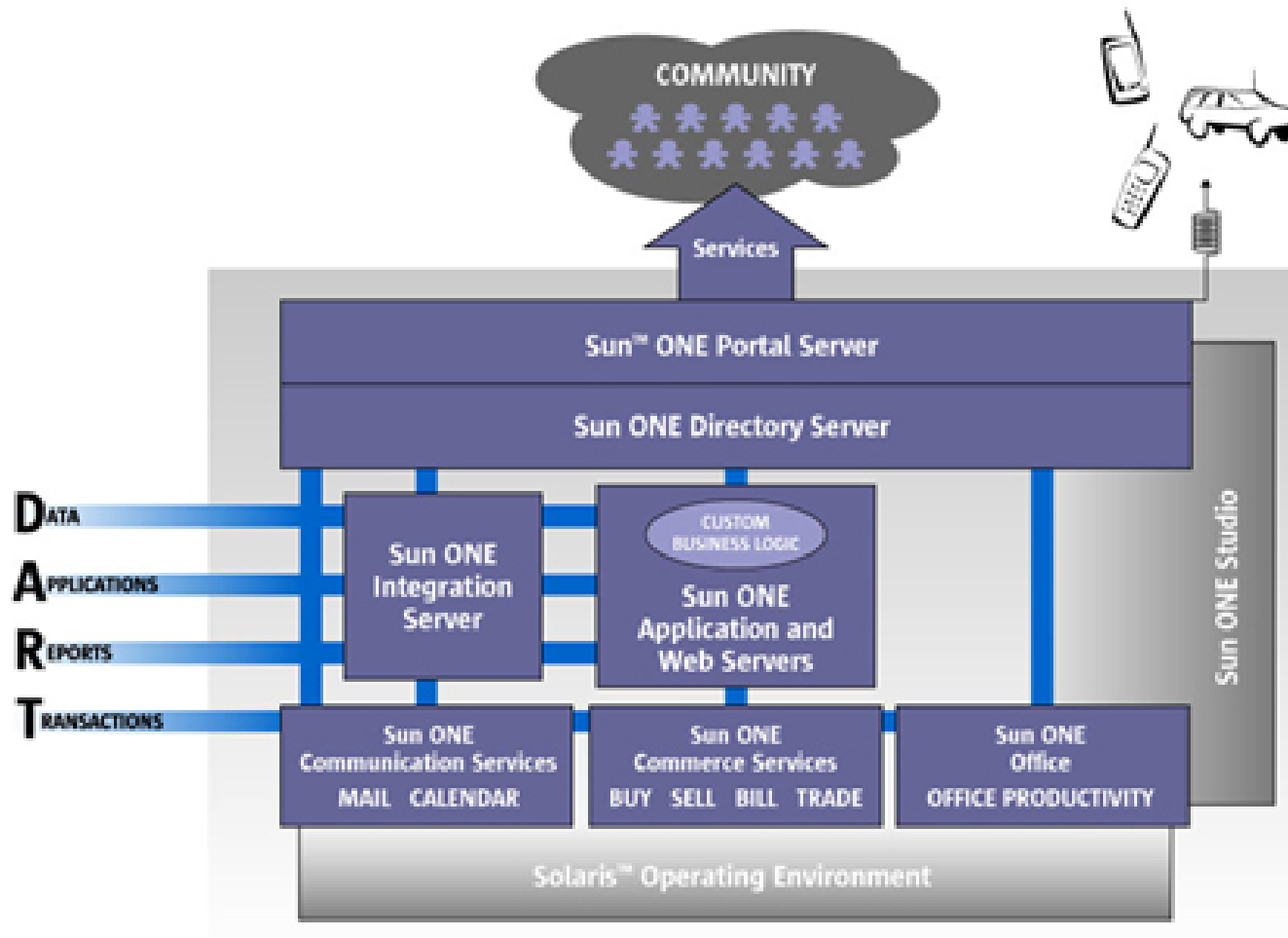
- MSFT product suite (no standard)
- Previous version was Windows DNA (included COM+, MSMQ, etc.)
- .NET framework replaces Windows DNA
- .NET servers (SQL server, BizTalk server, etc.)
- Hailstorm services/ My Services, Passport.NET
- Visual Studio.NET
- Common Language Runtime (CLR, provides language neutrality through IL code)
- New C# language

Sun ONE architecture

- “The Sun ONE architecture is Sun's software vision, architecture, platform, and expertise for **solving many of today's enterprise integration, interoperability, and development issues**. Based on the Java 2 Platform, Enterprise Edition (J2EE), it provides an easy evolution from nonintegrated enterprise applications to fully integrated and interoperable Web services.”

[Source: Web Services Journal
www.sys-con.com/webservices]

Sun ONE platform



<http://www.sun.com/software/sunone/index.html>

Sun ONE/ J2EE platform summary

- Sun Open Network Environment
- Industry standard
- 50+ vendors implement the standard (tools, application servers, etc.)
- Result of collaboration between vendors
- Based on JAVA technology – JRE interprets bytecode

J2EE and .NET Analogies

Feature	J2EE	.NET
Type of Technology	Standard	Product
Middleware Vendors	30+	Microsoft
Interpreter	JRE	CLR
Dynamic Web Pages	JSP	ASP.NET
Middle-Tier Components	EJB	.NET managed components
Database Access	JDBC, SQL/J	ADO.NET
SOAP, WDSL, UDDI	Yes	Yes
Implicit Middleware (load-balancing etc.)	Yes	Yes

Web Services: Introduction

The term "web services" has **two levels of meaning**

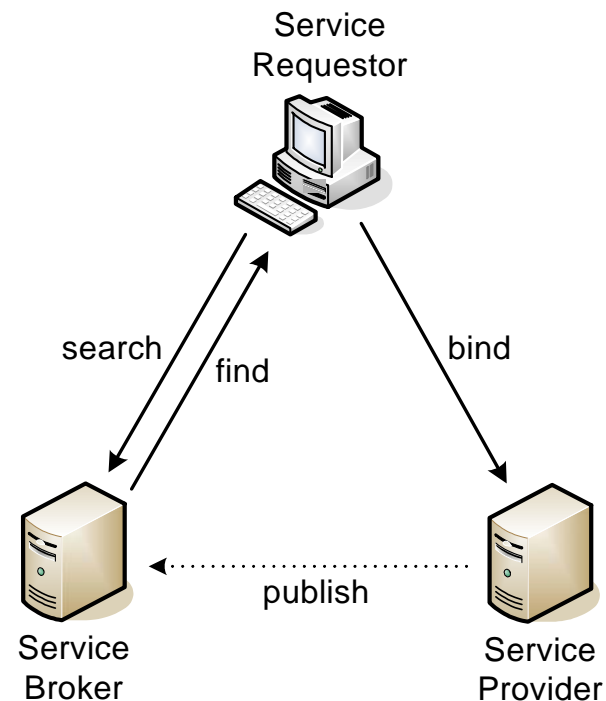
- Specifically, web services are a **stack of emerging standards** that describe a service-oriented, component-based application architecture.
- Conceptually, web services represent **a model** in which discrete tasks within e-business processes are distributed widely throughout a value net.
- Web Services: Principles
 - Object Orientation
 - Central Repositories
 - Internet Technology
 - Distributed Systems

Web Services: Introduction

- Working definition of W3C:
 - “A Web service is a software application identified by a URI, whose interfaces and binding are capable of being defined, described and discovered by XML artifacts and supports direct interactions with other software applications using XML based messages via internet-based protocols.”

- Service Oriented Architecture

- For more info:
<http://www.w3.org/TR/ws-arch/>



Types of Web Services

- 'There are two types of Web services – simple and complex. Simple Web services provide basic request/ response functionality, are typically not transactional in nature, and provide simple HTTP-S/SSL based security. They are developed supporting three primary Internet standards – SOAP, WSDL and UDDI.
- Complex Web services can be characterized as multi-party, long-running business conversations, that involve sophisticated security, such as non-repudiation and digital signatures, as well as business-to-business collaboration and business process management'

[Oracle 2001]

Web Service Standards: supported by...



... etc.

Web Services “Hype”

- Smart
- Vendors agree on all technologies
- Dynamic discovery
- Silver bullet

Web Services “Reality View”

- Smart? **Not yet**
- Vendors agree on all technologies? **Not yet**
- Dynamic discovery? **No**
- Silver bullet? **No**

Conclusion

- Web services are old technologies wearing a new hat
 - Get XML/ HTTP with sockets
 - Describe services with IDL or interfaces
 - Register services via Java Naming and Directory Interface or Active Directory
- Nothing mind-blowing here
- Lots of companies have been doing Web Services

Benefits of Web Services

- Loose application coupling
- Independent application evolution
- B2B—cheaper, leverages Internet
- EAI—non-intrusive integration
- “Component wars” and “Languages wars” do not affect interoperability
- All vendors are pushing for web services
- Some interoperability
- Standardisation of integration technologies
- Convenience APIs and tools

Web Services vs. RPC

Web Services

- programming language independent
- not (relatively) efficient space/time processing
- easily bound to different transport(s)
- firewall friendly
- synchronous RPC, or asynchronous messaging
- no binding to particular set of client/server frameworks (yet)

better suited to loosely coupled, coarse grain contracts

RPCs/ MOM

- typically bound to a particular (set of) programming language(s)
- efficient in space/time processing
- usually bound to a particular transport
- firewall unfriendly
- typically bound to client/server frameworks (e.g. COSS)

better suited to tightly coupled, fine grain contracts

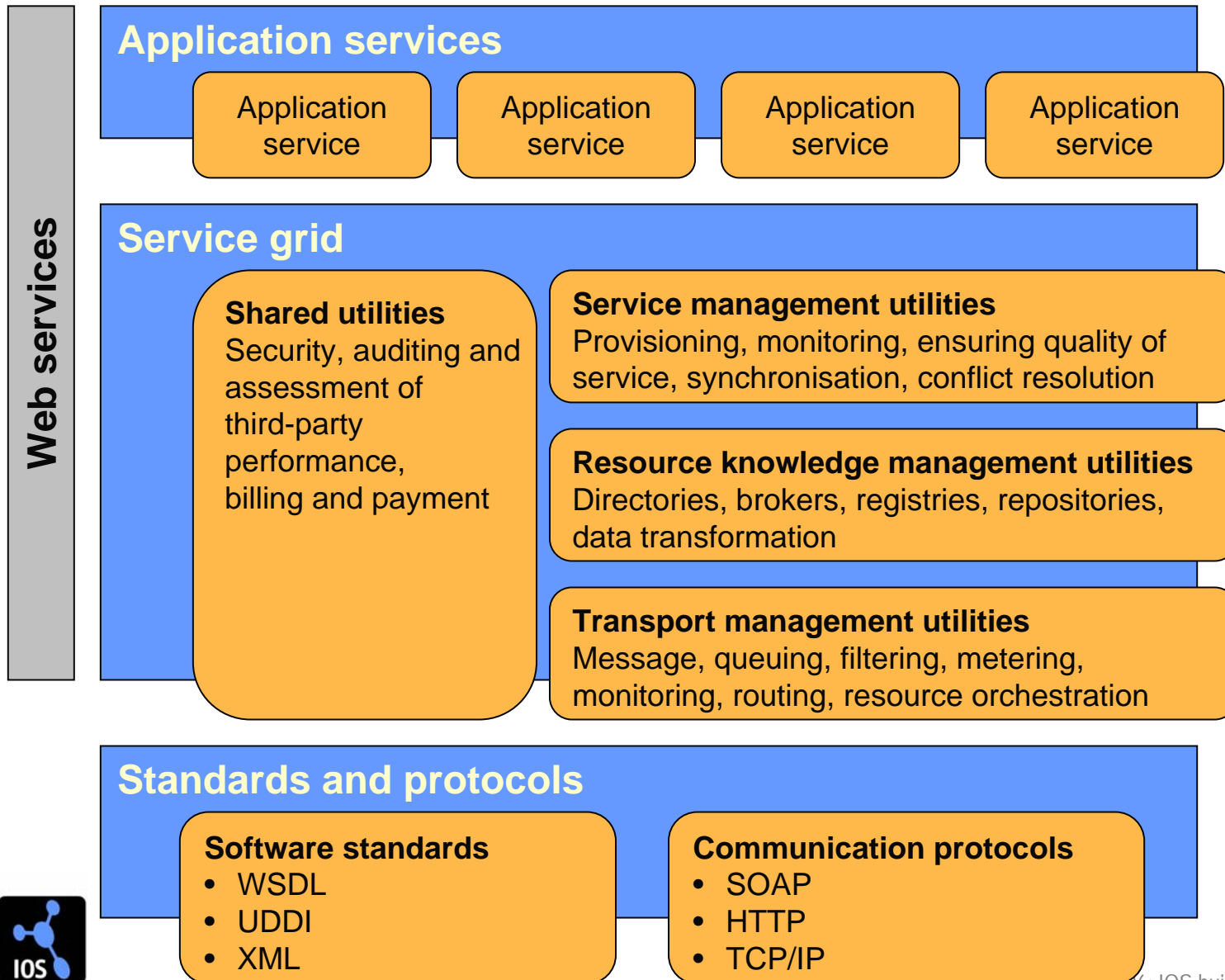
SOA - Service-oriented architecture

- SOA is an architectural style whose goal is to achieve loose coupling among interacting software agents
- A service is a unit of work done by a service provider to achieve desired end results for a service consumer
- Both provider and consumer are roles played by software agents on behalf of their owners
- In an SOA environment, nodes on a network make resources available to other participants in the network as independent services that the participants access in a standardized way

SOA - Service-oriented architecture

- SOAs comprise loosely joined, highly interoperable application services
- These services interoperate over different development technologies (e.g. Java, .NET), the software components become very reusable
- SOA provides a methodology and framework for documenting enterprise capabilities and can support integration and consolidation activities

The Web Service Architecture



Emerging and evolving protocols

- **XML-RPC** is a Remote Procedure Calling protocol that works over the Internet
- Simple Object Access Protocol (**SOAP**) provides a simple and lightweight mechanism for exchanging structured and typed information between peers in a decentralised, distributed environment using XML
- Universal Description, Discovery and Integration (**UDDI**) is a specification for distributed Web-based information registries of Web services
- Web Services Description Language (**WDSL**) is a specification to describe networked XML-based services
- **ebXML**: Business conversations, flow

(See W3C XMLP matrix for more details: <http://www.w3.org/2000/03/29-XML-protocol-matrix>)

SunONE/J2EE vs. .NET: Web Service Support

Sun ONE platform services and .NET services use common open standards such as those based on XML and SOAP. Therefore, the Sun ONE platform is able to use .NET Web services and is able to provide Web services to the .NET environment.

J2EE

- JAXP (Java API for XML parsing)
- RAD development tools through 3rd parties

.NET

- RAD development of web services through Visual Studio.NET
- No ebXML

References

- Frank, U.: Standardisierungsvorhaben zur Unterstützung des elektronischen Handels, in: Wirtschaftsinformatik 43 (2001) 3, pp. 283-293.
- Linthicum, D. S.: Enterprise Application Integration. Reading, Mass. (Addison Wesley) 1999.
- Pinkston, J.: The Ins and Outs of Integration, in: EAI Journal August 2001, pp. 48-52.
- Further Sources:
 - The Integration Consortium: <http://www.eaiindustry.org>
 - Business integration Journal: <http://www.bijonline.com>
 - W3C: Web Services Architecture: <http://www.w3.org/TR/ws-arch/>