Business Process Digitalization and Cloud Computing

2. Enterprise Systems Architectures

Andrea Morichetta, Phd

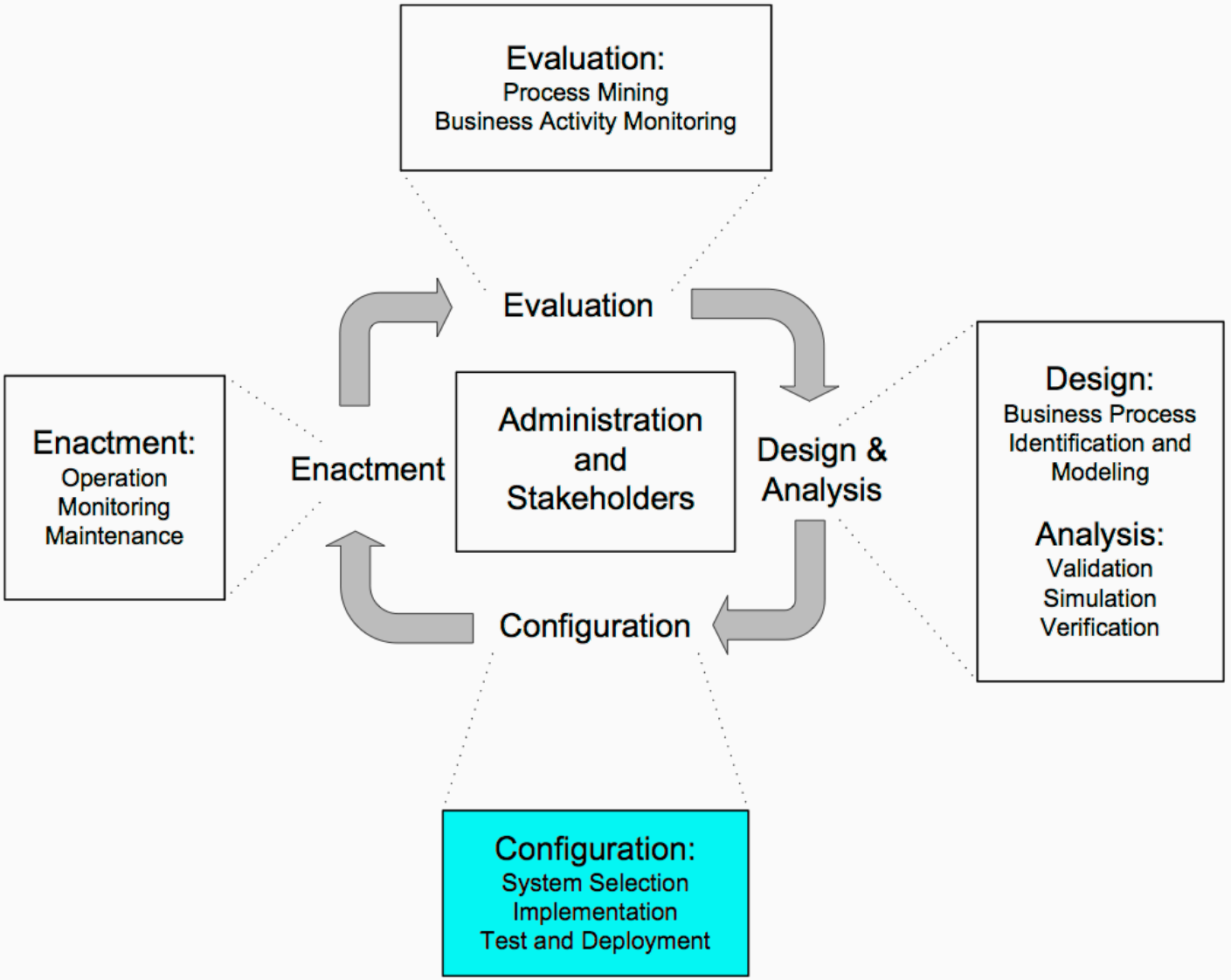
October 3, 2017

Computer Science Division

Table of contents

Enterprise application and their integration

Business Process Lifecycle: Configuration



Principles

Edsger Dijkstra principles:

Separation of concerns permit to handling the system complexity.

- **reuse:** subsystem can be used in different applications.
- **flexibility:** response to change, modified and exchanged.

Information hiding provide an interface which protects the communication with the program from its implementation

Software architecture

Software architecture play a central role in handling the complexity of software system.

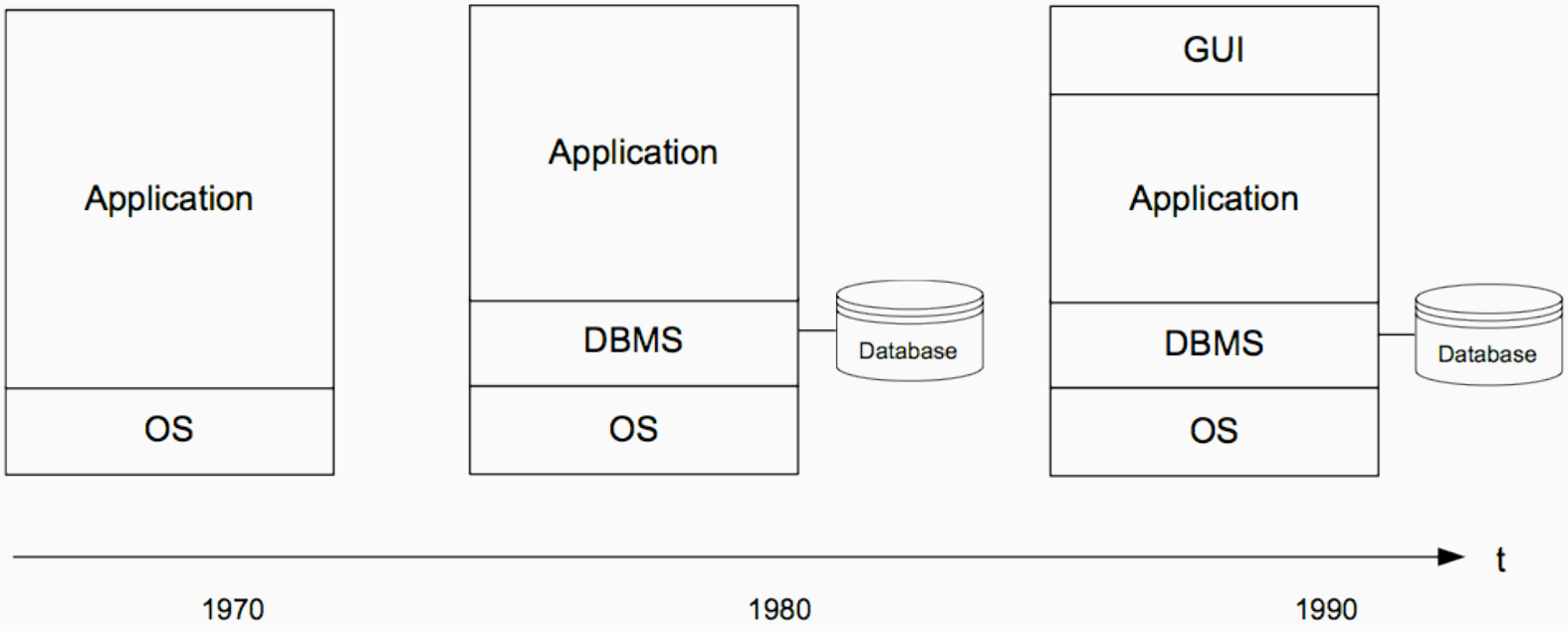
Software architecture

defines a structure that organizes the **software elements** and the **resources** of a software system

Software elements and resources

are represented by subsystems, with specific responsibilities and relationships

Early systems architecture



Programming interfaces (OS)

Relational Database

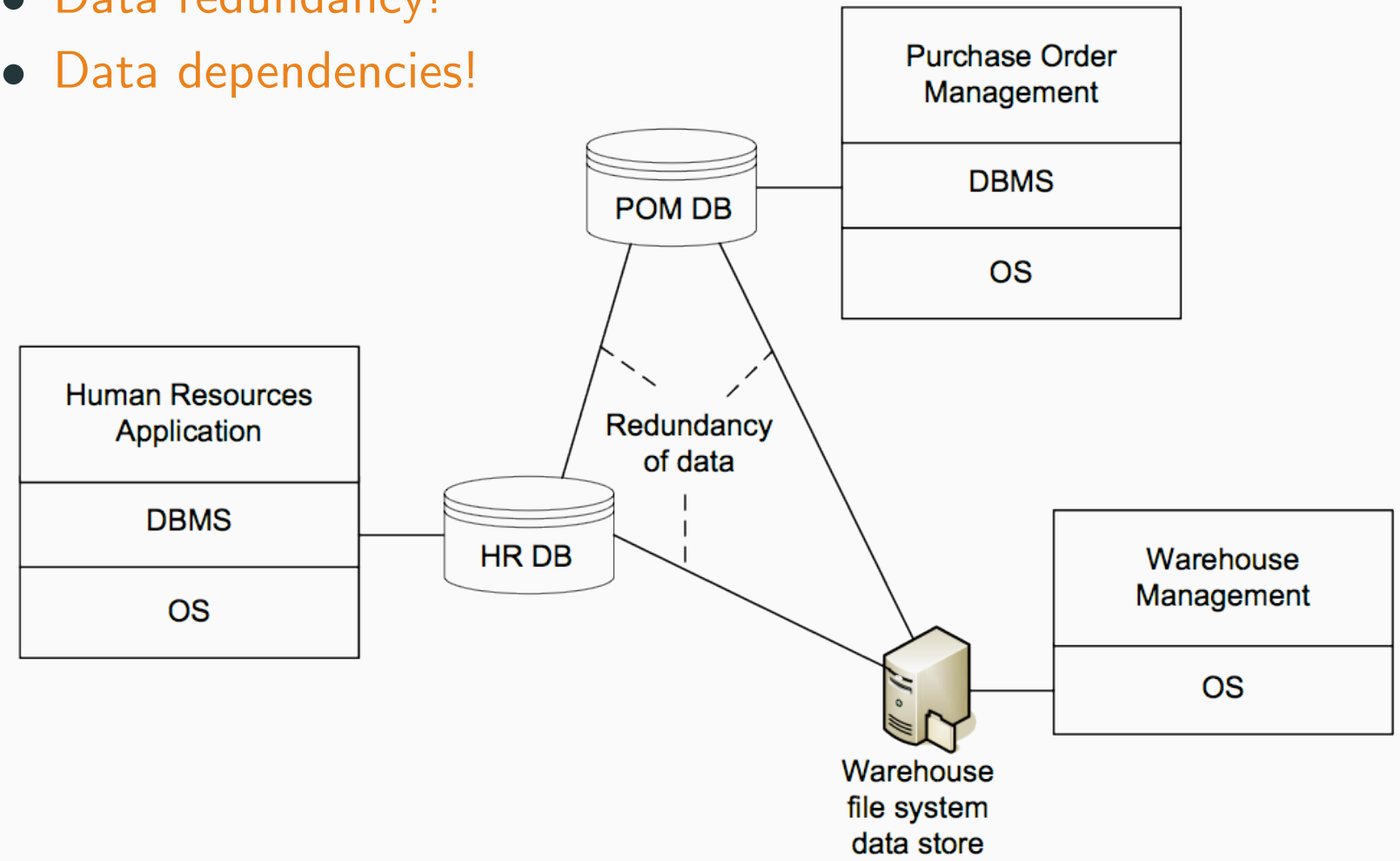
Advanced user interface

Changes

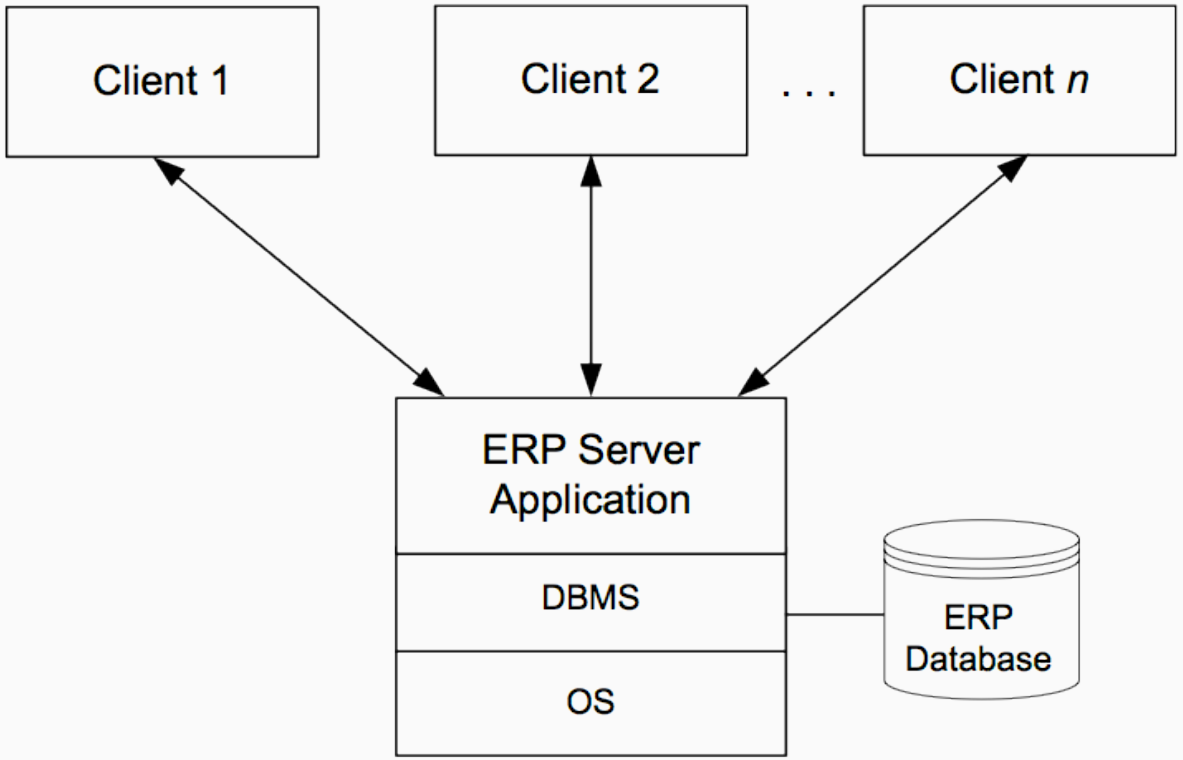
- Hard to implement
- Lack of dependency and replication
- Any modification of an application was a complex and error-prone activity, with domino effect (e.g. change of customer address format)

Enterprise systems

- Lack of Integration!
- Data redundancy!
- Data dependencies!



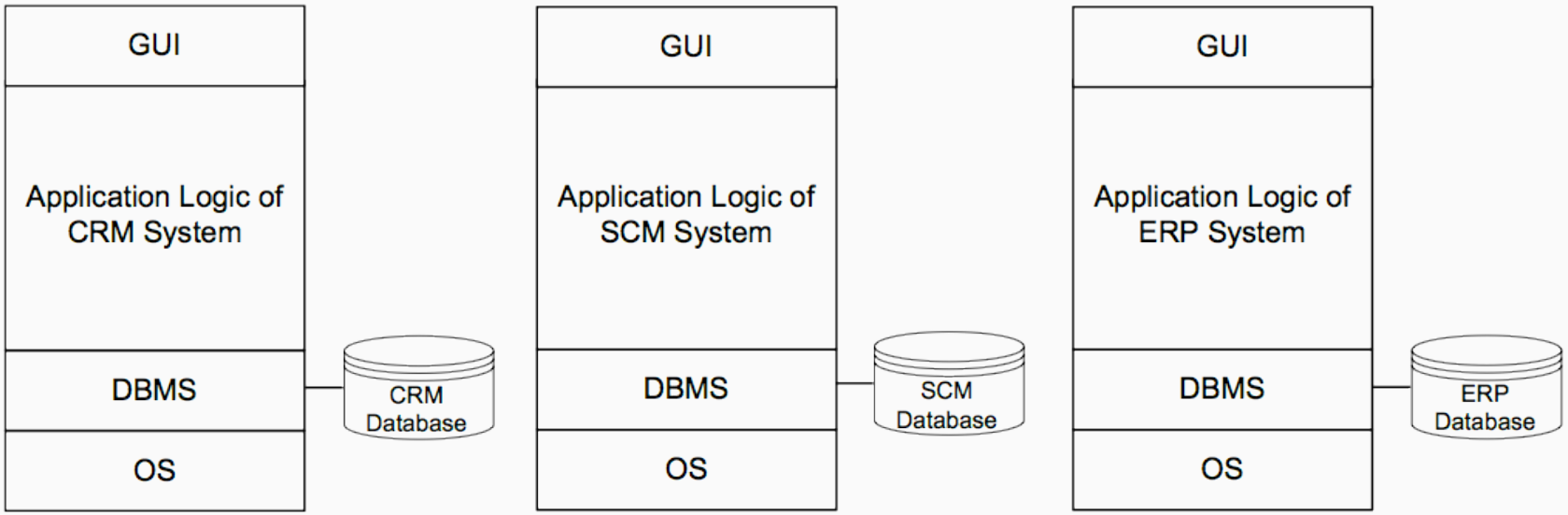
Two-tier Client-Server architecture



M. Weske: Business Process Management,
© Springer-Verlag Berlin Heidelberg 2012, 2007

- One centralized database
- Integrated server applications
- Remote data access

Siloed enterprise application



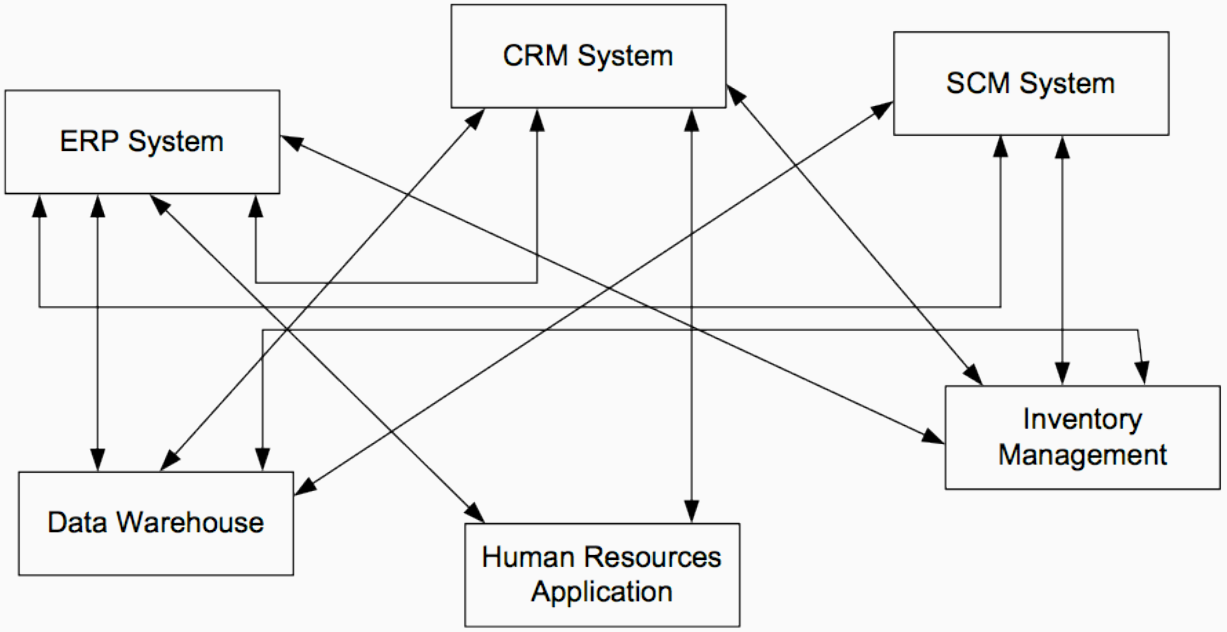
M. Weske: Business Process Management,
© Springer-Verlag Berlin Heidelberg 2012, 2007

- Indepent applicaitons
- Connected but not logically integrated

Enterprise Application Integration

- Enterprises are facing the challenge of integrating **complex software systems in a heterogeneous information technology** landscape
- Enterprise Application Integration is defined as the use of software and computer systems architectural principles to **integrate a set of enterprise computer applications**

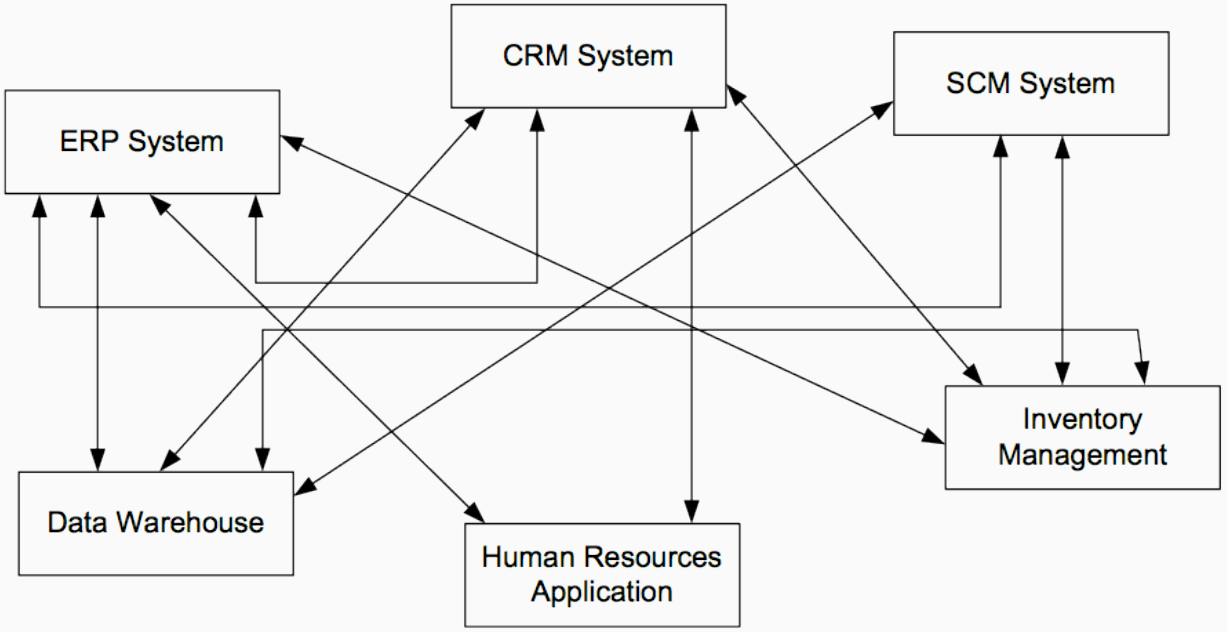
Point-to-Point integration



- Each integration project requires design and implementation
- Too many **interfaces** to develop $N \times N$
- How many **links**?

$$\sum_{i=1}^{N-1} i = ??$$

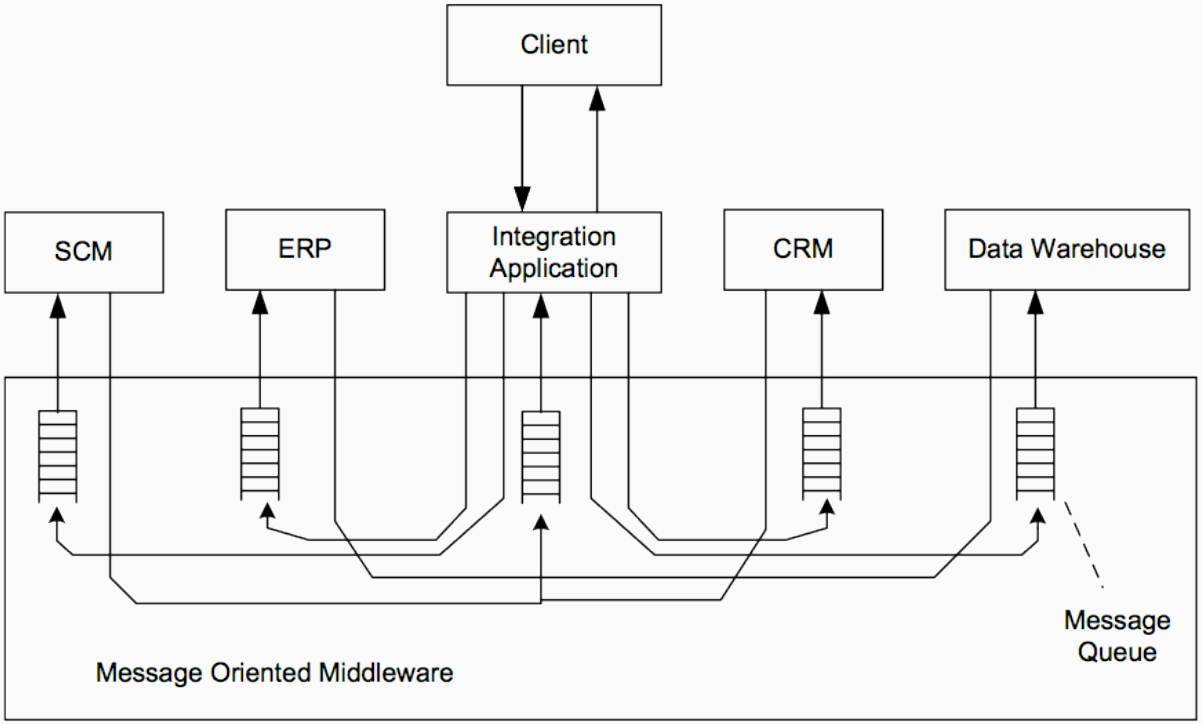
Point-to-Point integration



- Each integration project requires design and implementation
- Too many **interfaces** to develop $N \times N$
- How many **links**?

$$\sum_{i=1}^{N-1} i = \frac{N(N-1)}{2}$$

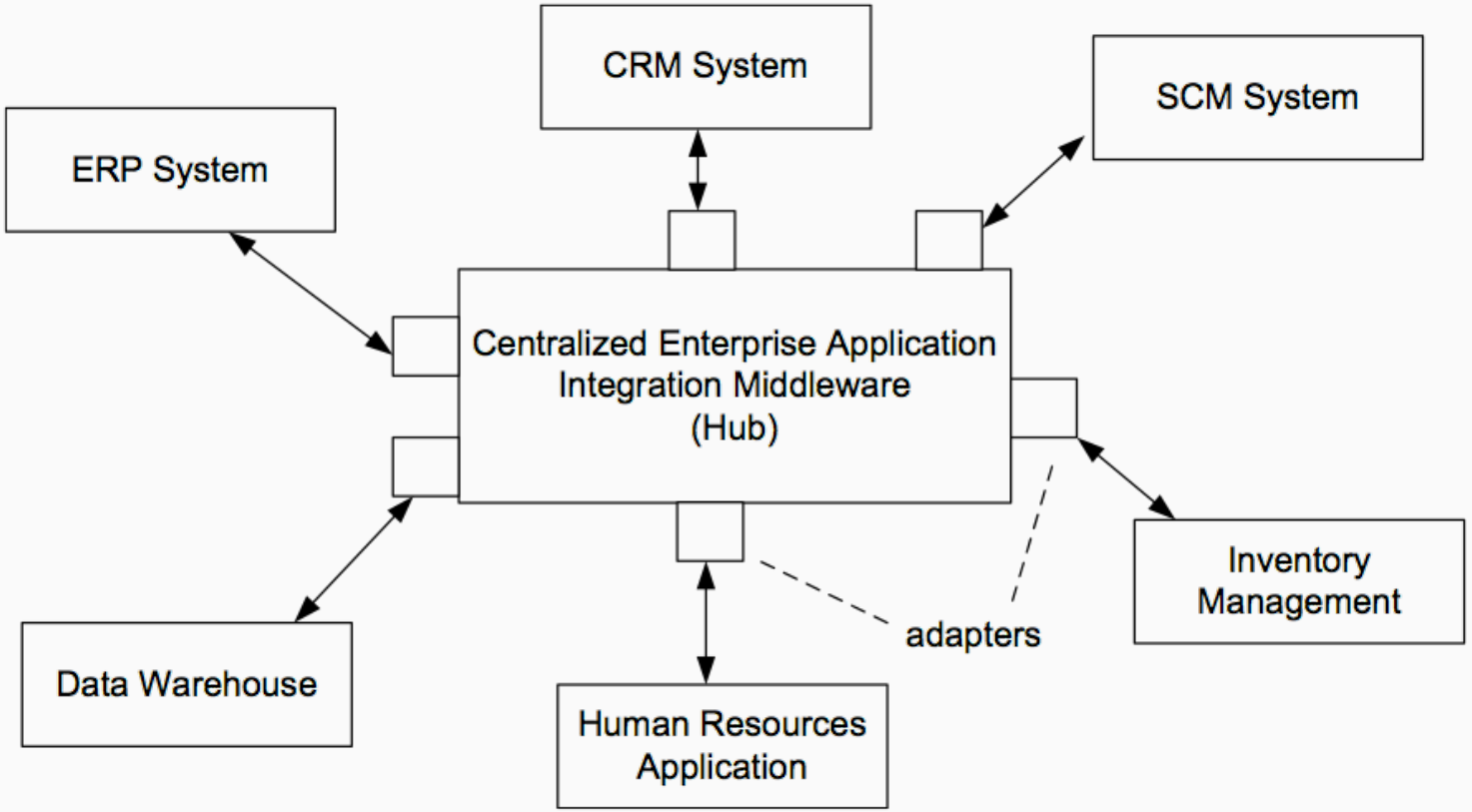
Message oriented middleware



M. Weske: Business Process Management,
© Springer-Verlag Berlin Heidelberg 2012, 2007

- Cooperation realized using the **integration application**
- process not directly connected
- Messages must be queued and enqueued
- Point-to-point connection in message oriented middleware

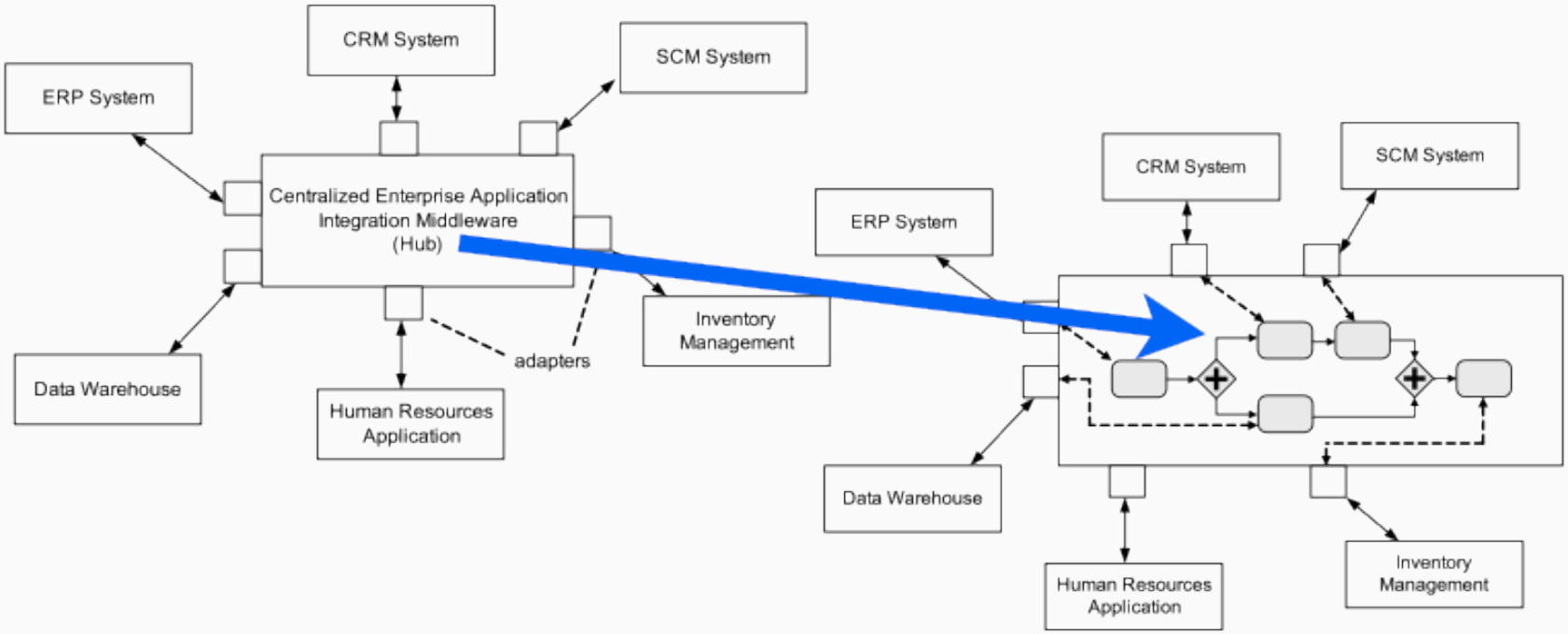
Hub-and-Spoke integration



M. Weske: Business Process Management,
© Springer-Verlag Berlin Heidelberg 2012, 2007

- Centralized hub
- Connection can be reduced
- How many link? N

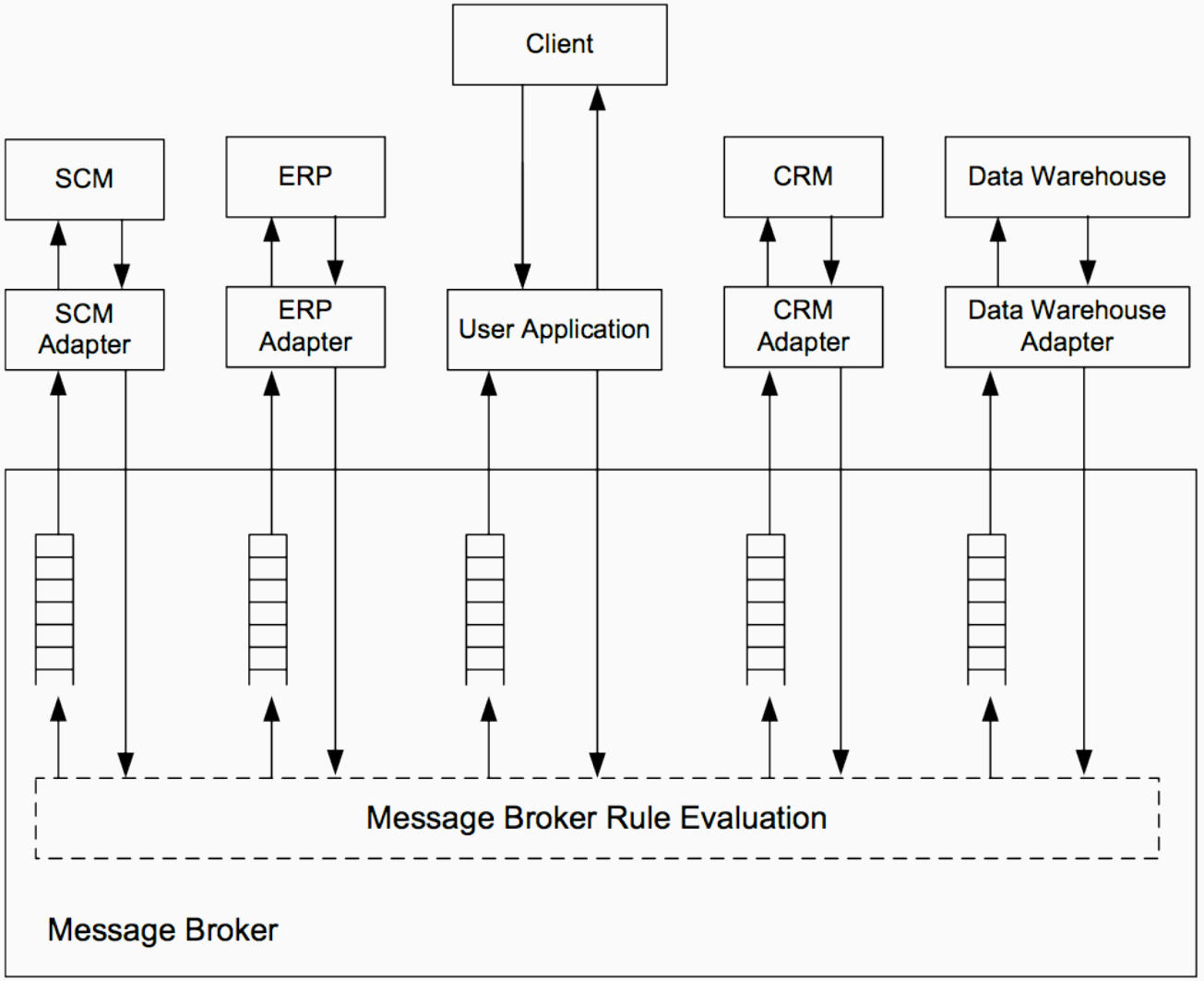
Hub-and-Spoke integration



Message brokers - Publish/Subscribe

- On a technical level, **message brokers** can be used to realize a hub and spoke enterprise application integration system
  - Message brokers are software systems that allow a user to define **rules** for communication between applications
  - Changes can be specified in a **declarative way** in the central hub, rather than by coding in the applications
  - The **queues** are used for guaranteed delivery of messages
- **Publish/subscribe** is a mechanism to link applications to message brokers
  - The idea is that applications can **subscribe to certain messages** or types of messages
  - Applications can also publish messages

Message Brocker integration



M. Weske: Business Process Management,
© Springer-Verlag Berlin Heidelberg 2012, 2007

- Uses declarative rules that de-couples senders from receivers

Problems!!

- The message broker contains **considerable application logic**
- This **application logic is hidden in the rules** that the message broker uses to relay messages
- Complex dependencies between rules can emerge, so that changing one rule might have undesired implications on the overall system behavior
- **Configuration and management of adapters** and message brokers can become cumbersome

Questions?