Middleware definitions and overview

Revision : 545

Chantal Taconet

September 2021

Overview

1. Which middleware?
2. Middleware for separation of concerns
3. Which middleware family
4. Architecture
5. Synthesis

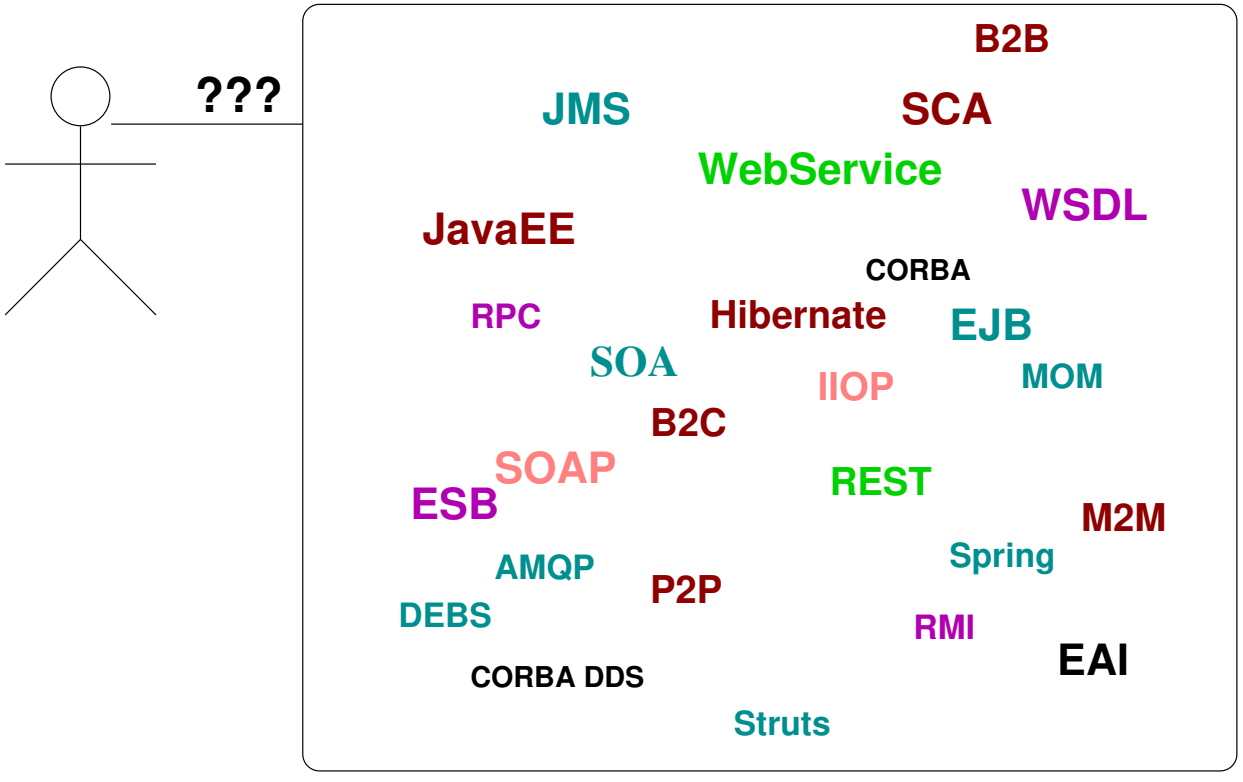
Middleware definitions

- Several definitions [1]
  - *Middleware is software glue.*
  - *Middleware is the slash in Client/Server*
  - *Software that mediates between an application program and a network.*
  - *Middleware is computer software that connects software components or applications. It is used most often to support complex, distributed applications. It goes on to say that it describes a piece of software that connects two or more software applications so that they can exchange data.*
  - *Middleware is any software that allows other software to interact.*
  - *Middleware is sometimes called plumbing because it connects application and passes data between them.*
  - *Middleware is software used for coupling high level system components (application) with basic system components (data and network)*

1. Found on `http://www.middleware.org/whatis.html` and `http://en.wikipedia.org/wiki/Middleware`

Middleware as a universal adapter to build high level applications ?

Which middleware?

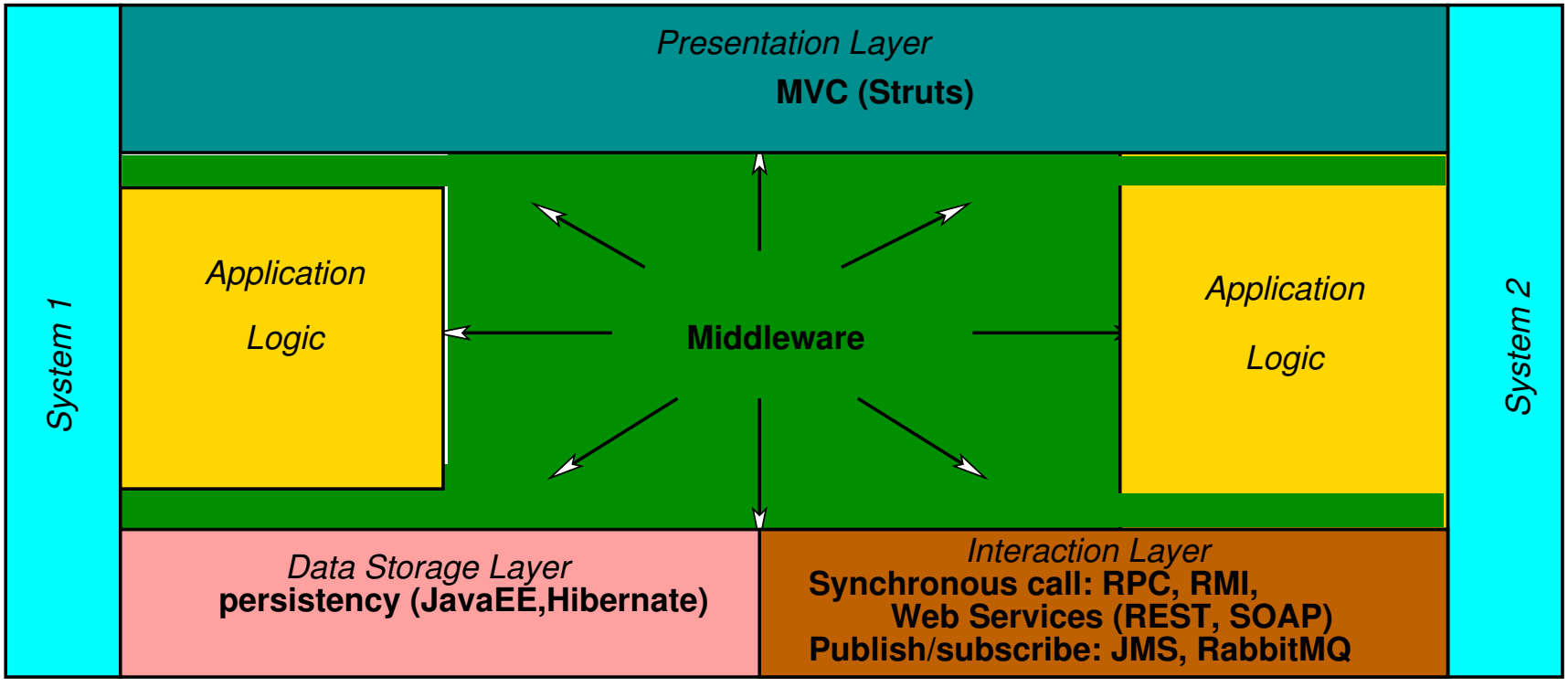


- A wide number of middleware technologies are hidden under those acronyms!

- To master the complexity: Understand the abstractions, classify the middleware

2 Middleware for separation of concerns

2.1 Middleware: several concerns



2.2 Separation of concerns and middleware

- Middleware is a solution for the separation of concern paradigm

In computer science, separation of concerns (SoC) is a design principle for separating a computer program into distinct sections, such that each section addresses a separate concern.

  - Separation of concern enables application designers to focus on their business
    - Use standard middleware components for handling non business preoccupations

- Through middleware, separation of concern is reached for :
  - Heterogeneity
  - **Distribution** of pieces of software
  - Persistency of components
  - New middleware for new preoccupations (*e.g.*, middleware for the IoT)

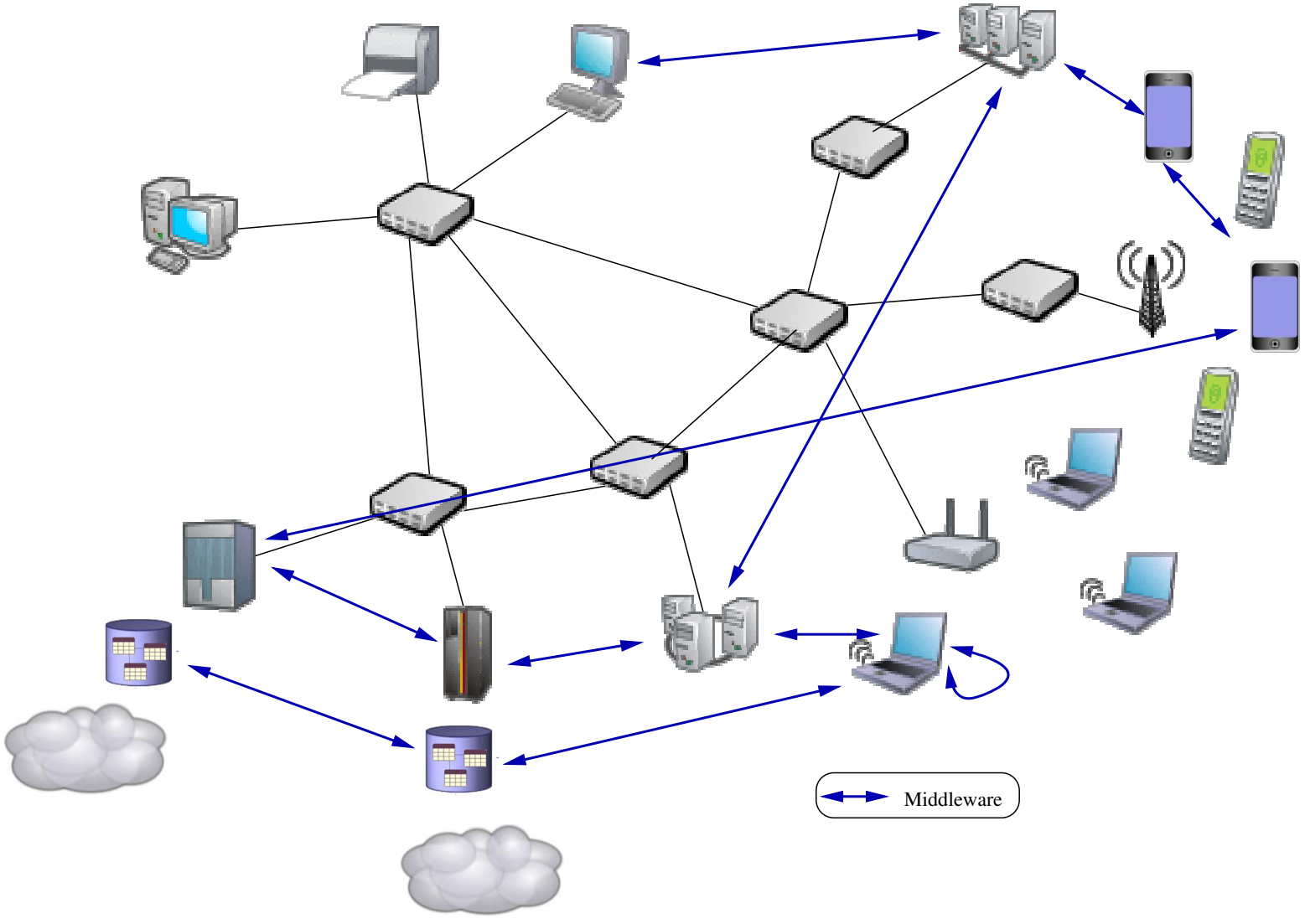
2.3 Levels of heterogeneity addressed by middleware

- Middleware may address several level of heterogeneity
  - Hardware heterogeneity (*e.g.*, *Little Endian* and *Big Endian* representation)
  - Operating System heterogeneity (*e.g.*, library availability)
  - Language heterogeneity (*e.g.*, one piece of software in *C*, another piece of software in *java*
  - Application logic heterogeneity (*e.g.*, data transformation from one application to the other)

2.4 Middleware for several levels of distribution

- Pieces of software connected by middleware may be distributed on:
  - Several processes (in the same computer)
  - Several computers (in the same local area network)
  - Several networks (in the same company)
  - Several companies

2.5 Examples of software distribution

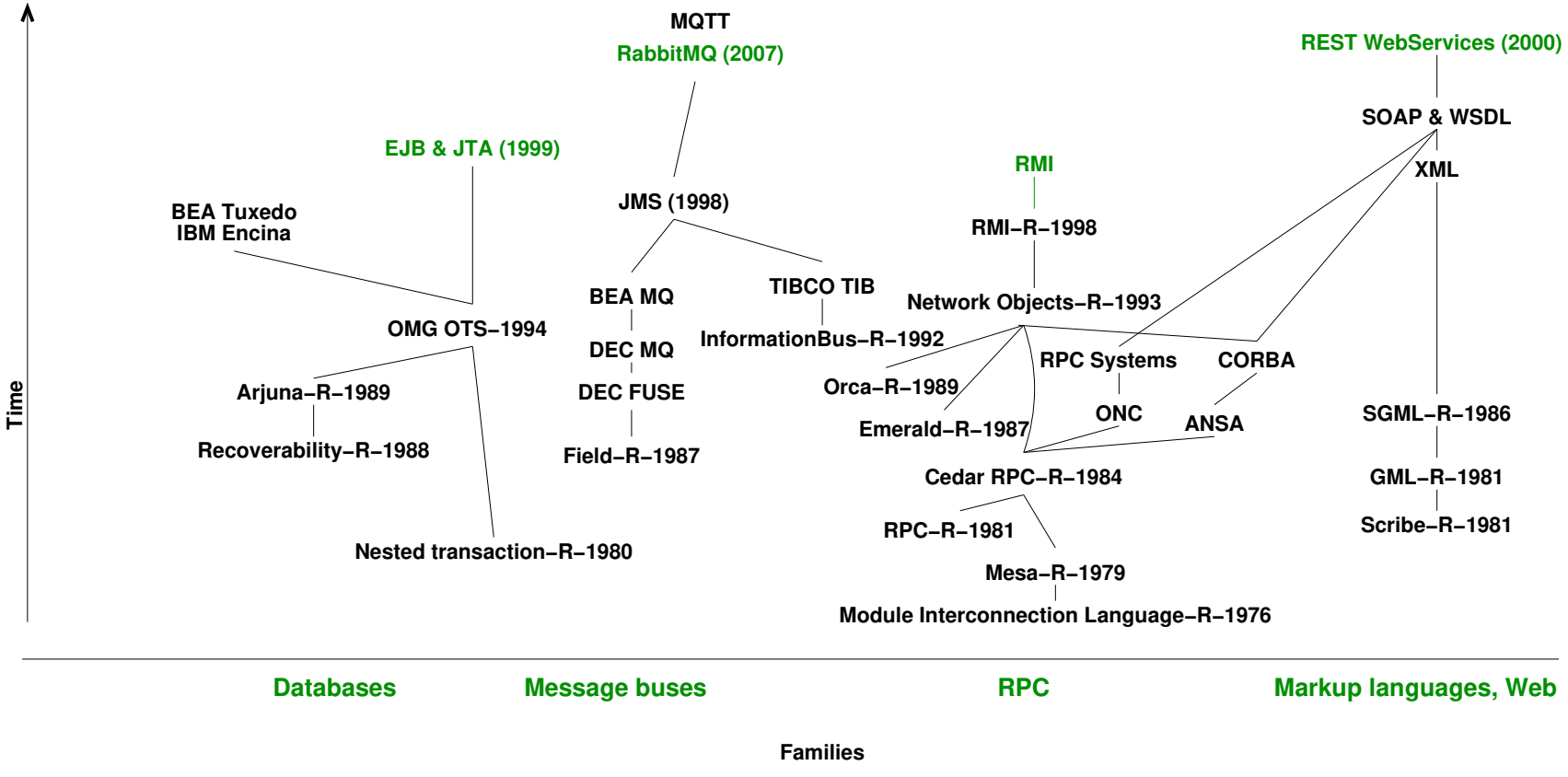


3 Which middleware family

3.1 Families of middleware

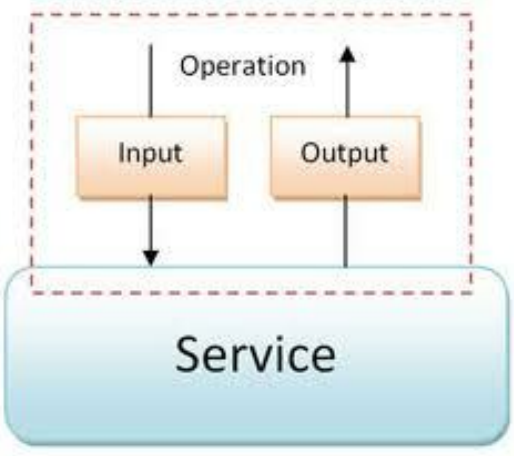
- RPC middleware
- Object Middleware
- Message Oriented Middleware
- Component Middleware
- Service Middleware
- Database middleware
- Persistency middleware

3.2 Main family history



3.3 Interaction styles: synchronous call

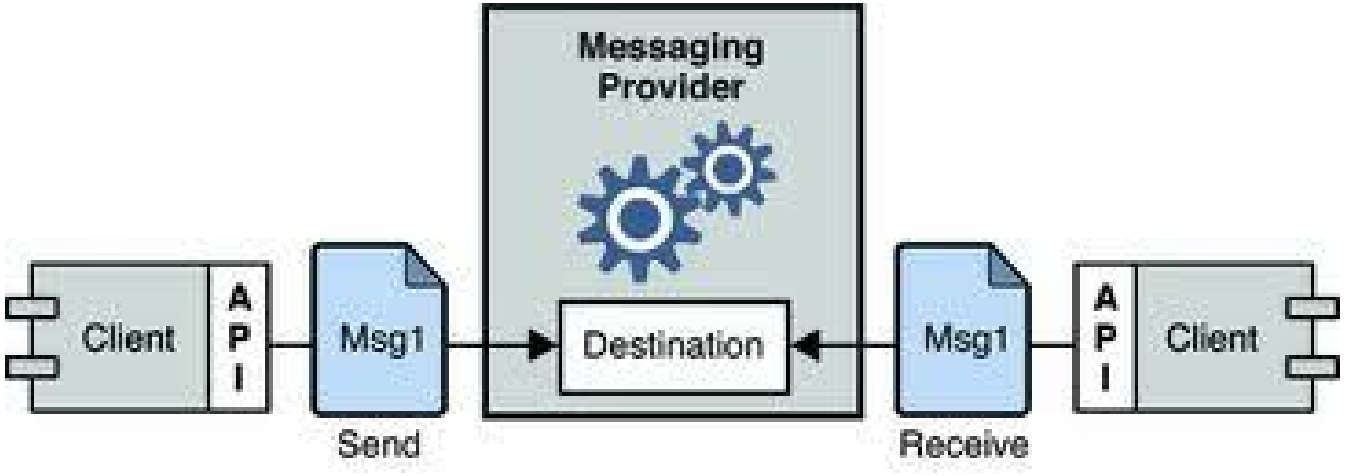
- Middleware for distributed **synchronous calls** (interaction layer)



- **RPC**: request broker,
- **CORBA**: object request broker, multi-languages, Local Area Network (LAN)
- RMI: object request broker, java, LAN
- Web services (synchronous messages): multi-languages, Wide Area Network
  - REST (microservice architecture)
  - SOAP (Service Oriented Architecture, service orchestration)

3.4 Interaction styles: publish/subscribe

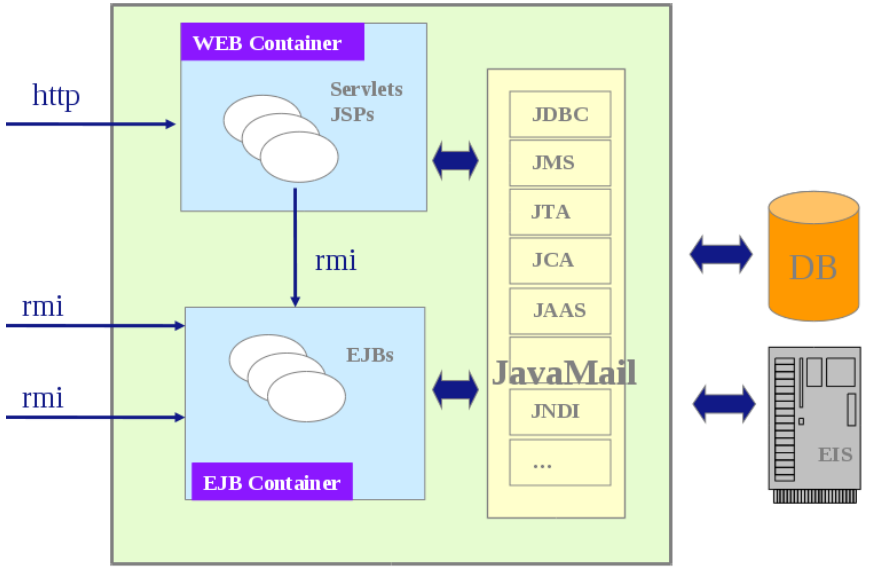
- Middleware for **publish/subscribe** (interaction layer)



  - **MQTT** For the IoT
  - **JMS** LAN
  - AMQP, RabbitMQ

3.5 Object/Service/Component lifecycle : servers and containers

- **Application server** manager: instantiation, containers
  - Application Servers
    - JavaEE (JBoss, glassfish, Websphere):
    - Light servers : **Spring**
  - Web container : **Web Server (tomcat, jetty, LiteWebServer)**:



3.6 Data management

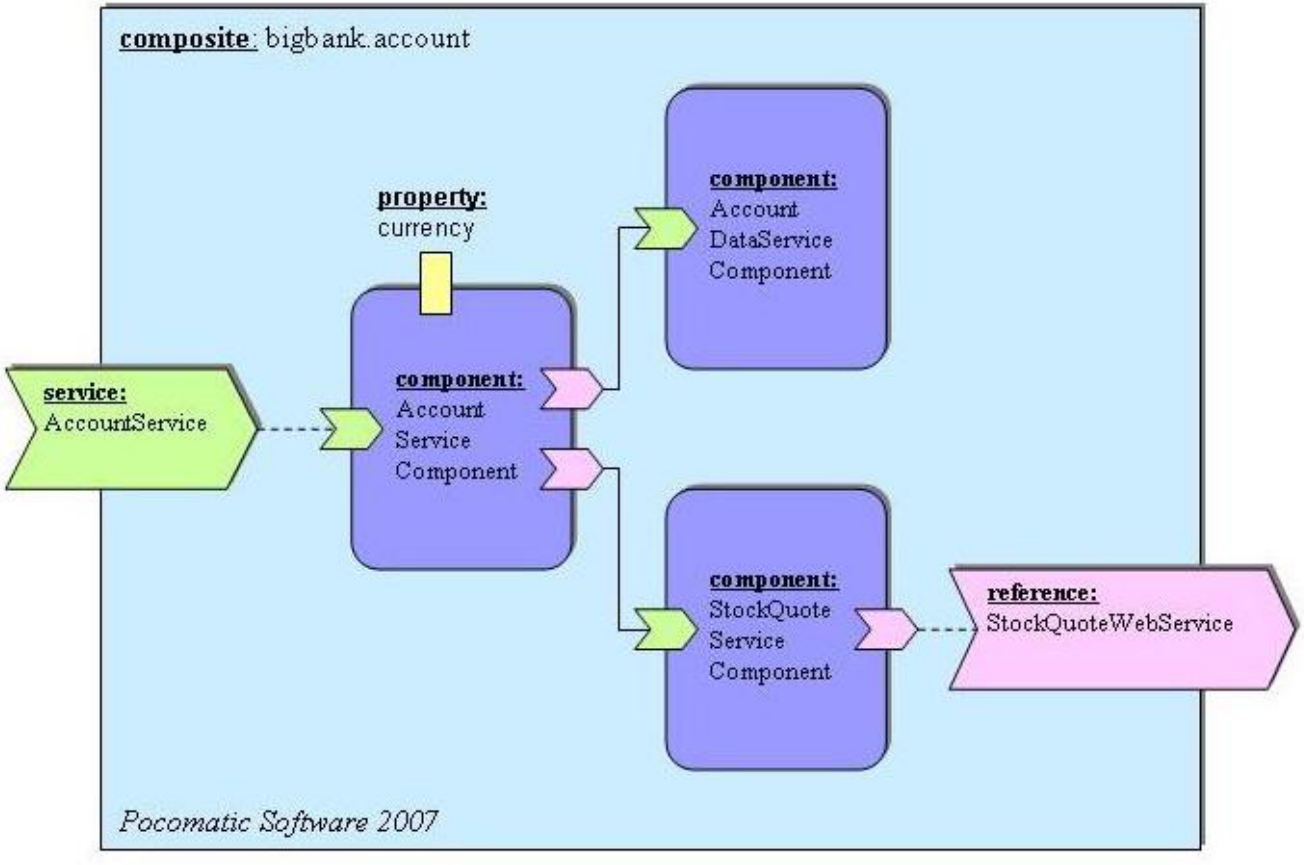
- **Persistency middleware** handles persistency of data or objects (data layer)
  - JavaEE (EJB) includes persistency preoccupation (various technologies)
  - **Hibernate** is a persistency framework (from object to relational database paradigm)

4 Architecture

4.1 3 tiers Architecture

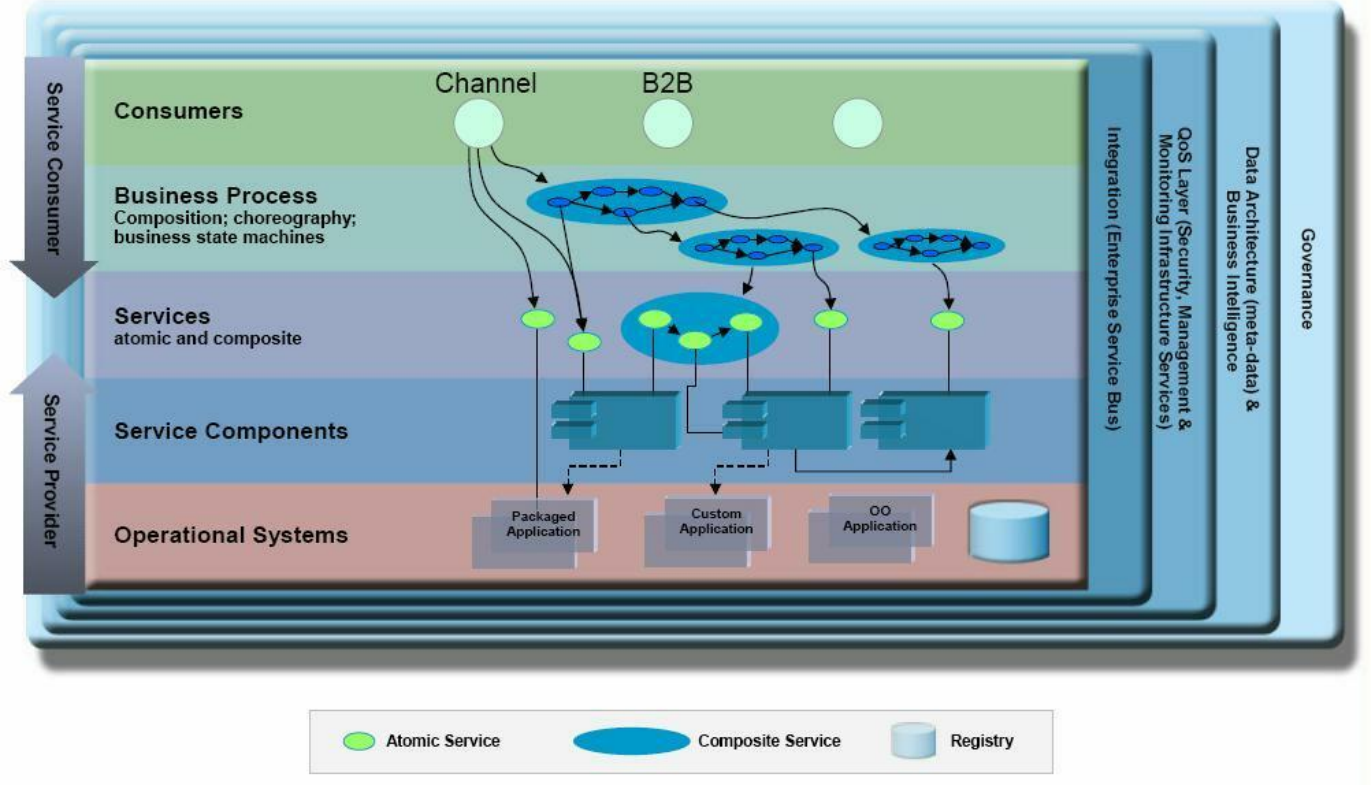
4.2 Component based Architecture

- Component abstraction
- Component Assembly (*e.g. SCA Service Component Architecture, Fractal components, CORBA Component Model*)



4.3 Service Oriented Architecture

- **Service Oriented Architecture** (for sequence of services)
  - Service abstraction
  - Service Orchestration

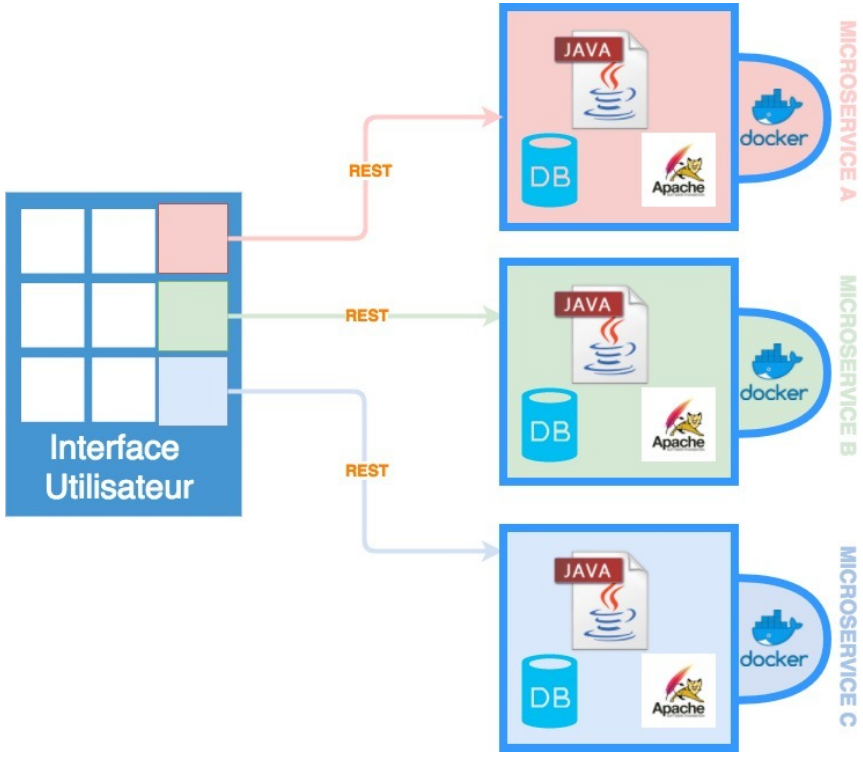


2

2. Source de la figure https://www.ibm.com/developerworks/mydeveloperworks/blogs/devaprasad/

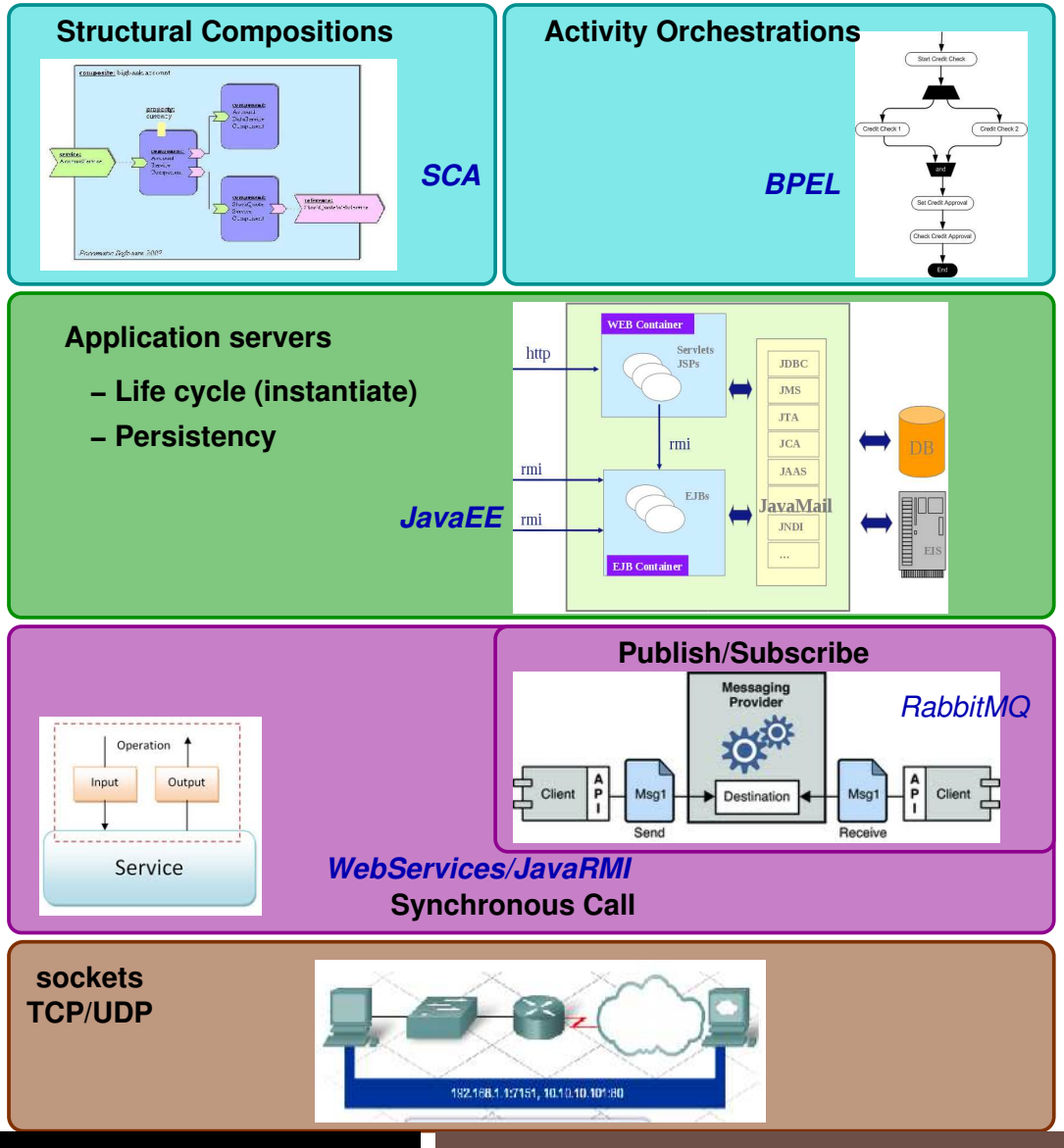
4.4 Microservice architecture

- A microservice is a software architectural style that structures an application as a collection of loosely coupled services.
- Advantages:
  - modularity
  - continuous delivery
  - better scalability
- Microservices interaction patterns
  - Services in a microservice architecture are often processes that communicate over a network
    - For synchronous interactions: REST over HTTP (one of the most popular)
    - For Asynchronous interactions: AMQP and Akka actors are good candidates



5 Synthesis

5.1 Layer view



5.2 Conclusion

- In the design of a distributed application, you first choose the middleware family, you choose the middleware itself later on (e.g. you first choose **Synchronous interaction** style and then *Java RMI* or *REST* WebService).
- Middleware connect pieces of software implemented separately (by different companies, developers . . .) and available on the network.
- Standardisation is essential to connect pieces of software
- Universal adapter is of course not possible:
  - Many technologies are available with different characteristics (*e.g.*, target platform, semantics, efficiency)
  - The basic of middleware is about distribution (RPC, RMI) sometimes called plumber solutions.
  - Above distribution, higher abstractions may be built: publish/subscribe, data distribution, persistency, presentation, naming, workflow, orchestration and composition.