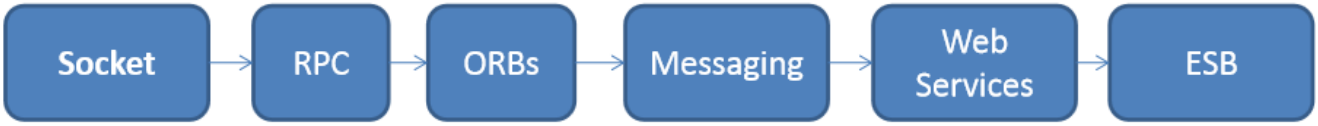
Evolution of software integration

Following figure shows the six important method of software integration.



Integration based on sockets

- Socket ::= IP address + (TCP/UPD) port
- Real-time data transfer
  - binary data transfer but can be normal text or XML as well
  - no direct method sharing (can be done by hand)
  - TCP and UDP connections are possible. UDP is 3 times quicker but one-way communication
- Persistent or On-Demand communication channel
  - because of connection time-loss usually persistent channels are better, but periodically “ping” messages should be sent. (in order to avoid connection closing). In case of any problems reconnection is possible.
  - in case of UDP channels an extra TCP channel is available for synchronizing
- Results in the fastest possible transmission.
  - Where the number of transactions per second up to ~ 20 transactions, there should have been applied. (50ms / sec transfer)

RPC Remote Process Call

- Classical client-server programming. The underlying socket solution remains hidden from developers.
- The channel is not only suitable for simple data transfer, remote execution of procedures also natively supported (on API level).
- Interface description
  - It is necessary to describe the remote procedures with some form of XDR (External Data Representation Standard) can help.
- Platform independence
  - Can be solved by the help of cross-platform interface descriptors

Object Request Broker

- First introduced to object-oriented programming paradigm in software integration. It extends the concept of objects in the “distant objects”.
- Distant objects communicate with each other
  - they can share both data and methods
- Known implementations: Corba and Java-RMI

- Marshalling/Unmarshalling concepts appear. Platform independent serialization of the transferred parameters (or data)
- Language independent interface description methods with Corba IDL (interface definition language)
- Initial concept of Service Registry.
    - services are browse-able in an uniform manner.
- Despite its antiquity is still supported in the modern application servers (like JBoss, IBM Websphere)

Messaging systems

- Messaging systems are asynchronous parallel systems
- In the background there is socket communication as well
- The system is asynchronous because we should not wait for the answer, execution flow is continuous, non-blocking.
- Each function call creates a message on the “Message Queue”. Message processing is alway parallel in a different process.
- This indirect method facilitated the loose coupling of different systems.
- Guaranteed message delivery is feasible, because of the intermediate message queue.
- Synchronous function calls can be simulated with a second message queue.

Web services

- XML (eXtendible Markup Language) based communication, which combine the advantages of RMI, CORBA, Messaging.
- Messages and responses are text messages in XML format.
- Advantage: communication can be followed easily, because data streams are not binary.
- Disadvantage: XML processing is more resource intensive than binary data processing.
- Interface description is in XML format as well. WSDL (Web Service Definition Language), is programming language, platform independent.
- UDDI (Universal Description, Discovery and Intergration) It is a registry for discovery, describe services.
- SOAP (Simple Object Access Protocol) is applied for data transmission.
    - Simpler solution is the XML-RPC. (XML-Remote Process Call)

Enterprise Service Bus (ESB)

- The Web services are not effectively solve the interconnection of heterogeneous systems.
- Not resolve the communication protocol deviations, i.e. during communication, the protocol offered by the service is different than the protocol used by the client.
- Protocol and message transformation is also possible
    - message routing – based on context and content
    - QoS (Quality of Service) – performance, security, reliability
    - data enrichment – information extension with automatic extra data

EAI Requirements

EAI (Enterprise Application Integration) covers every part of an enterprise system including business processes, architecture, hardware, software.

- **Business Process Integration (BPI)**. It is very important for a corporation to specify the processes involved in the interchange of enterprise information. “This allows organizations to streamline operations, reduce costs and improve responsiveness to customer demands.” This can include process management, process modeling and workflow. Here, we involve the combination of tasks, procedures, organizations,required input and output information, and tools needed for each step in a business process.
- **Application Integration**: Here, the goal is to “bring data or a function from one application together with that of another application that together provide near real-time integration.” This can include, business-to-business integration, customer relationship management (CRM) systems which can be integrated with a company's backend applications, web integration, and building web sites that interact with multiple business systems.
- **Data Integration**: If we want the above two integrations to succeed, we must also integrate the data involved. Its location must be identified, recorded, and a metadata model must be built (a master guide for various data stores). Now, data can be shared or distributed across database systems, providing it is in a standard format such as COM+/DCOM, CORBA, EDI, JavaRMI, and XML.
- **Platform Integration**: Finally, the separate needs of the heterogeneous network must be integrated. Platform Integration deals with the processes and tools that are required to allow these systems to communicate, both optimally and securely, so data can be passed through different applications without difficulty.