Service-oriented Architectures: A Review

Reference

- Text book: Enterprise SOA: Service-oriented Architecture Best Practices, D. Krafzig, K. Banke and D. Slama, Prentice-Hall Inc., 2007.
- WS and SOA

SOA and WS

- A Service-Oriented Architecture (SOA) is a design model for linking computational resources, data and applications to perform services and deliver results to service consumers.
- Web Service (WS) standard provides a platform-independent method for messaging-based interaction of applications.

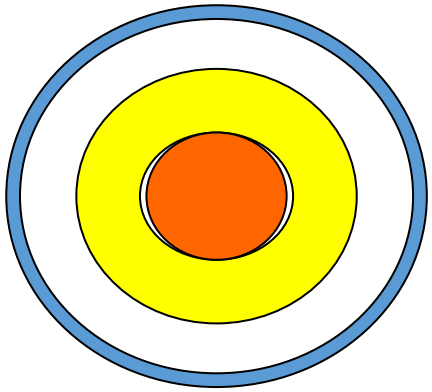
SOA Principles

- Loosely coupled (service provider and service consumer are loosely coupled: why?)
- Large scale: complex system with high level of heterogeneity and redundancies.
- Decoupling of functionality and technology
- Service contract and agreements
- Discoverability
- On demand composability of services: composite services concept
- Agility: respond to changes quickly
- Statelessness
- Inherent interoperability
- Standards
- Reusability

Evolution of the service concept

- A service is a meaningful activity that a computer program performs on request of another computer program.
- Technical definition: A service a remotely accessible, self-contained application module.

  -- From IBM



Object / Class

Compo nent

Service

Business Computing

- File systems to main frames
- Emergence of new paradigms such as Enterprise Resource Planning (ERP) and Supply Chain Management (SCM) placed complex requirements on the computing machines and applications.
- This was followed by huge compute (IT) demands for Enterprise Application Integration (EAI) and Enterprise Data Integration (EDI).
- An appealing characteristic of SOA is that it aligns these business entities by directly mapping them to services, thus enabling an enterprise integration on the business level, not on the technical level.

On to more fundamental concepts: Synchrony

- Synchronous and asynchronous communications
- Synchronous:
  - immediate response of communicating partners
  - Server process/thread blocks until response is completed
  - Follows request/response pattern
  - Used when servers are available all the time
  - Typically communicating partners are tightly coupled
  - Examples:
    - request from web client to a web browser for “search” or for “information”
    - CORBA procedure invocation
    - Java RMI (remote method invocation)
    - Traditional remote procedure call (RPC)

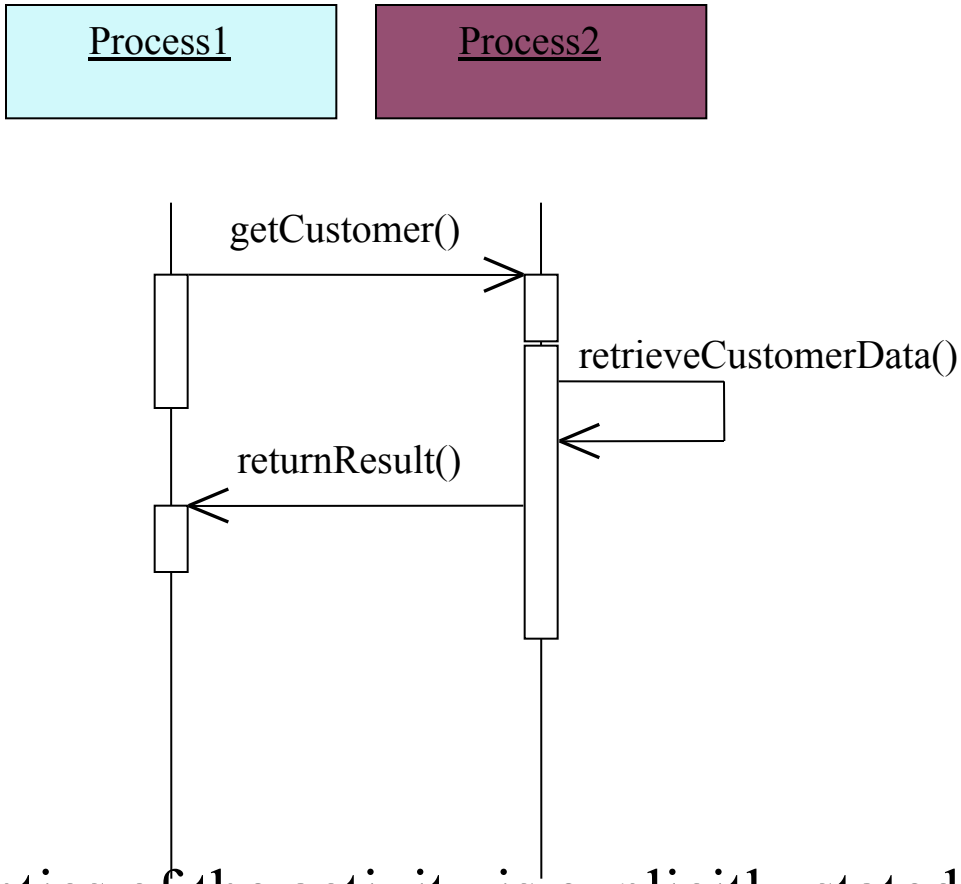
Asynchronous communication

- Communicating partners are decoupled
- Message driven:
  - sender creates a message and delivers it to a mediator who then sends it to “a” recipient
  - Server need not be available all the time
  - Sender and receiver loosely coupled
  - Can facilitate high-performance message-based system
  - Example:
    - Any event-driven system
    - Any messaging system (instant messenger)
    - Publish-subscribe mode communications

Interface vs Payload Semantics

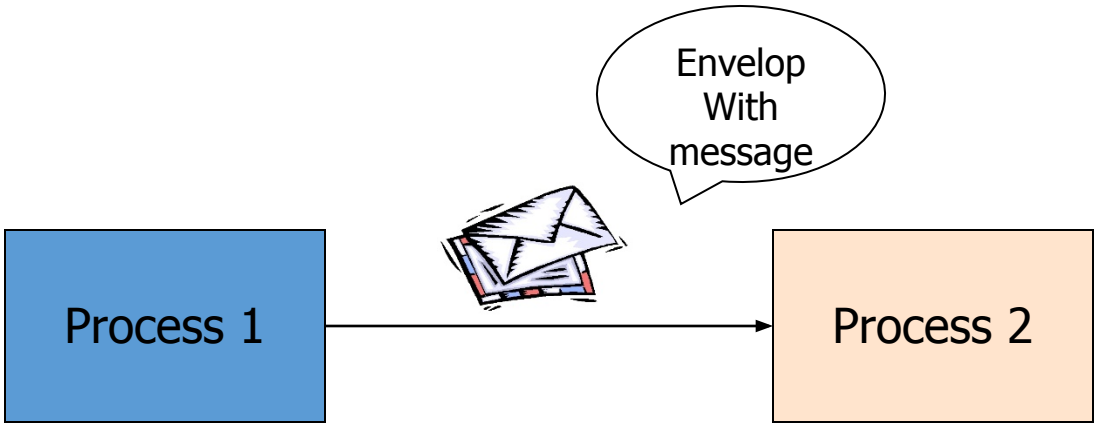
- Typically interaction between a client and a server results in the execution of an activity (ot transaction)
- Request needs to be specified by the request.
  - Interface semantics: Requested activity can be encoded in the operation signature in the server's "interface" or
  - Payload semantics: It can be embedded in the message itself

Interface Semantics



Semantics of the activity is explicitly stated in the message/method call

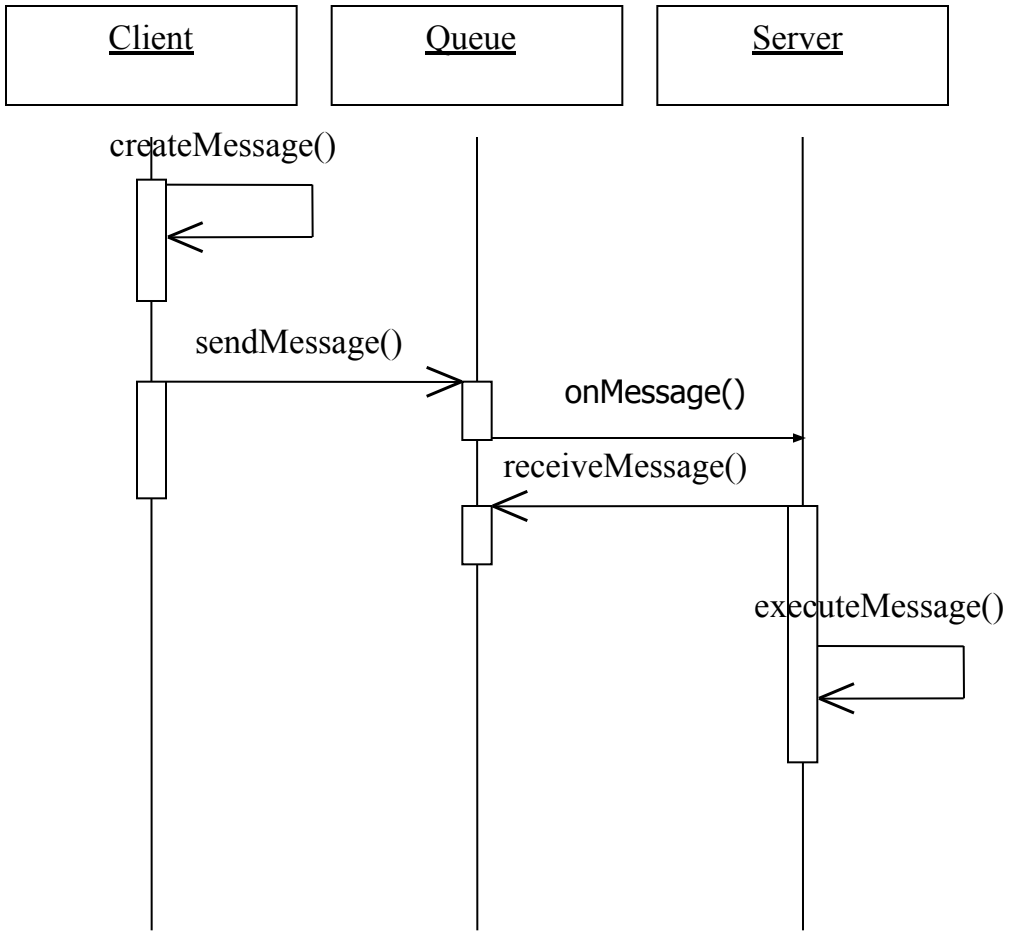
Payload Semantics



Requested transaction/activity is embedded in the message

Details of the activity not explicit; the semantics are embedded in the message

Payload Semantics



Payload semantics is generic

String transferMoney (amt: decimal, accTo: String)

{ …}

String executeService (message: String)

{ …}

Tight vs. Loose Coupling

- An important characteristics of an SOA that is a loosely coupled system.
- On the technology front this is driven by dynamic discovery and binding enabled by Universal Description, Discovery and Integration (UDDI)
- On the business front loose coupling addresses the growing need for companies to be flexible and agile with respect changes in their own processes and those of their partners
- How does loose coupling help in improving agility, flexibility and performance?

Tight vs. Loose coupling

Level	Tight coupling	Loose coupling
Physical coupling	Direct physical link required	Physical intermediary
Communication style	synchronous	asynchronous
Type system	Strongly typed (interface semantics)	Weak type system (payload semantics)
Interaction pattern	OO-style navigation of complex object trees	Data-centric, self-contained messages
Control of process logic	Central control of process logic	Distributed logic components
Service discovery and binding	Statically bound services	Dynamically bound services
Platform dependencies	Strong OS and programming language dependencies	OS- and programming language dependent

Service-oriented architecture (1)

- From “The new language of business : SOA and Web 2.0” by S. Carter, IBM Press, 2007
- Service-oriented architecture is a business driven IT architectural approach that supports integrating a business as linked, repeatable business tasks or services.
- It helps
  - innovation by assuring IT systems can adapt quickly.
  - increase flexibility of business processes
  - strengthen underlying IT architecture
  - reuse their existing IT investments by creating connections among disparate applications and information sources
- The above in turn help address increasing complexity, need for lowering development, integration and maintenance cost and obtain sustainable competitive edge through technology.
- SOA begins with a service that could be a simple business task such a checking the credit rating of a potential customer.

Service-oriented Architecture (2)

- From “Service-oriented architecture: A planning and implementation guide for business and technology”, by E.A. Marks, and M. Bell, Wiley & sons, 2006.
- SOA is a conceptual business architecture where business functionality, or application logic, is made available to SOA users or consumers, as shared, reusable services on an IT network.
- Services in an SOA are modules of business or application functionality with exposed interfaces, and are invoked by messages.
- Essential ingredients of an SOA are: services, enabling technology, SOA governance and policies, SOA metrics, organizational and behavior model (culture).

Service-oriented Architecture (3)

- From “ Service-oriented architecture: concepts, technology and design”. By T. Erl, Prentice-Hall Inc., 2005.
- Service-oriented architecture is a term that represents a model in which automation logic is decomposed in to smaller, distinct units of logic called services.
  - Collectively these units comprise a larger piece of business automation logic. These pieces can be distributed.
  - Services are autonomous units; messages are used for communication among these.
- Principles of SOA: loose coupling, service contract, autonomy, abstraction, reusability, composability, statelessness, discoverability

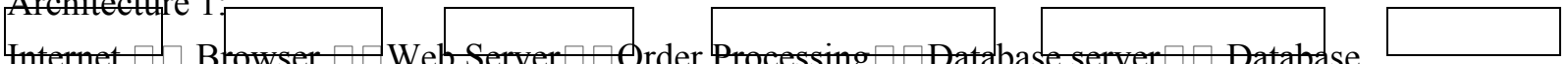
Service-oriented architecture (4)

- From “Service-oriented Architecture (SOA) compass: business value, planning and enterprise roadmap”. N. Bernstein, S. Bose, M. Fiammante, K. Jones and R. Shaw, IBM press, 2006.
- A service-oriented architecture is a framework for integrating business processes and supporting IT infrastructure as secure, standardized components– services– that can be reused and combined to address changing business priorities.
- Loose coupling, reuse, interoperability between systems.
- SOA is a synonym for solution architectures making use of Web service technologies such as SOAP, WSDL, and UDDI. Any product and project conforming to the WC3 Web services architecture (WSA).
- SOA is a set of business, process, organizational, governance and technical methods to enable an agile, business-driven IT environment for greater competitive advantage.

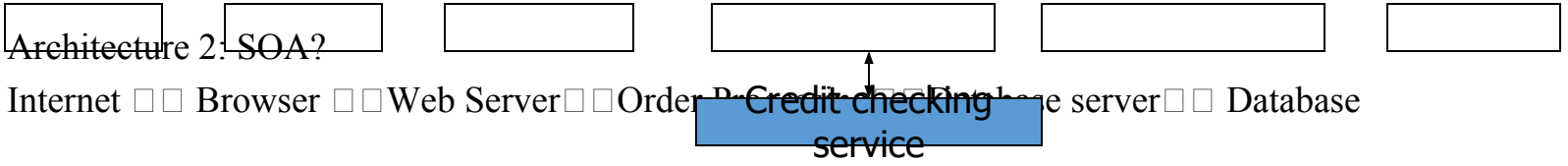
Service-oriented architecture (5)

- From "Service-oriented architecture for dummies", by J. Hurwitz, R. Bloor, C. Baroudi, M. Kaufman, Wiley & sons., 2007.
- Architecture implies thoughtful planning according to set of guidelines or rules. Ex: a house, a mall, Taj Mahal or Noah's ark
- Software architecture describes the overall design and structure of a computer system.
- In a service oriented architecture, business services interact with each other in ways similar to how various services of the restaurant interact.
- Basic architecture of an order processing system and an SOA of a the same. Lets analyze this further.

Architecture 1:



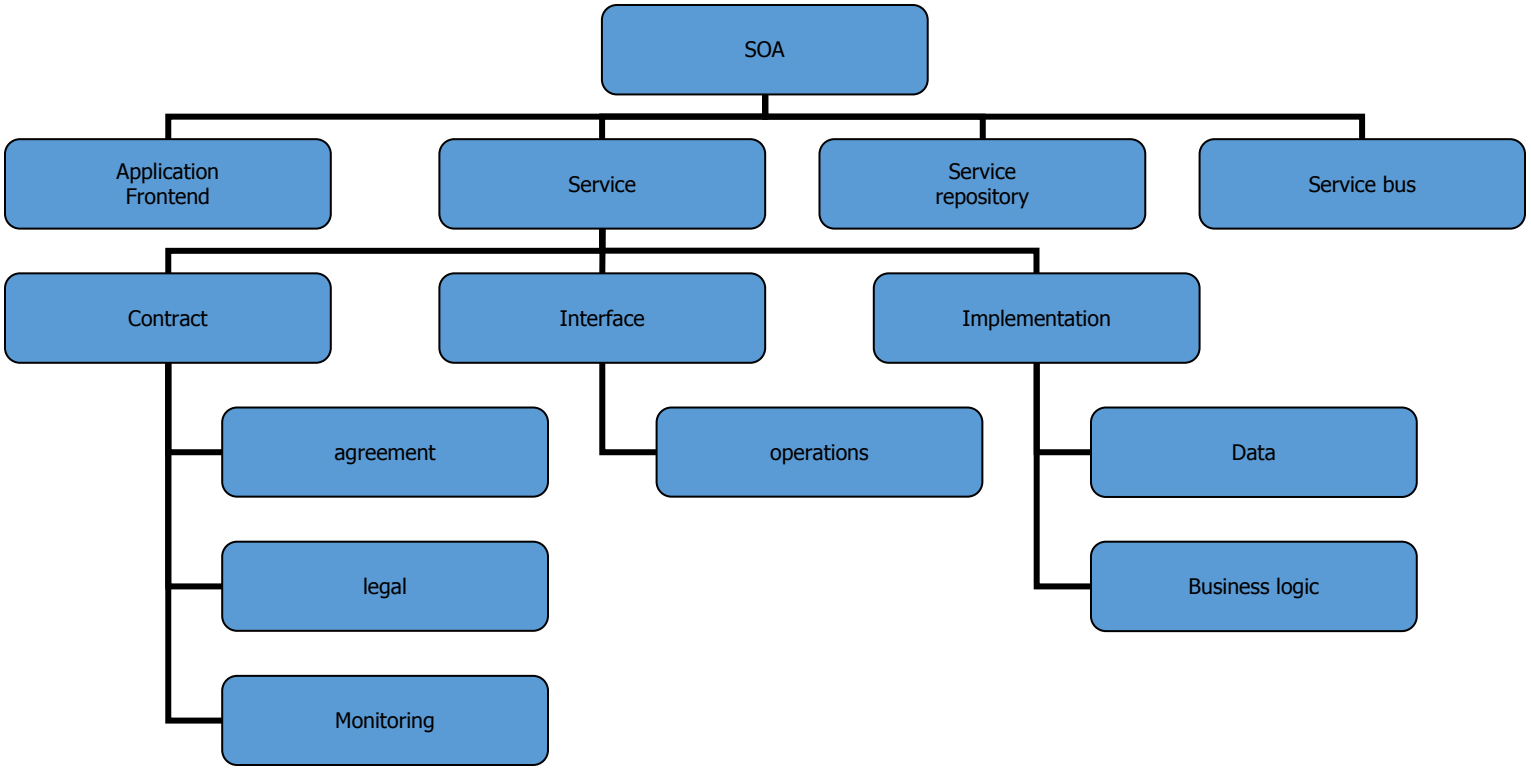
Architecture 2: SOA?



Service-oriented architecture (6)

- From “Enterprise SOA: Service-oriented architecture best practices” by D. Krafzig, K. Banke, and D. Slama, Prentice-Hall Inc., 2007.
- A software architecture describes software components of a system and assigns the functionality of the system to these components. (p.56)
  - It describes the technical structure, constraints, and characteristics of the components and the interfaces between them.
  - The architecture is the blueprint for the system and therefore high-level plan for its construction.
  - Lets look at example: web architecture

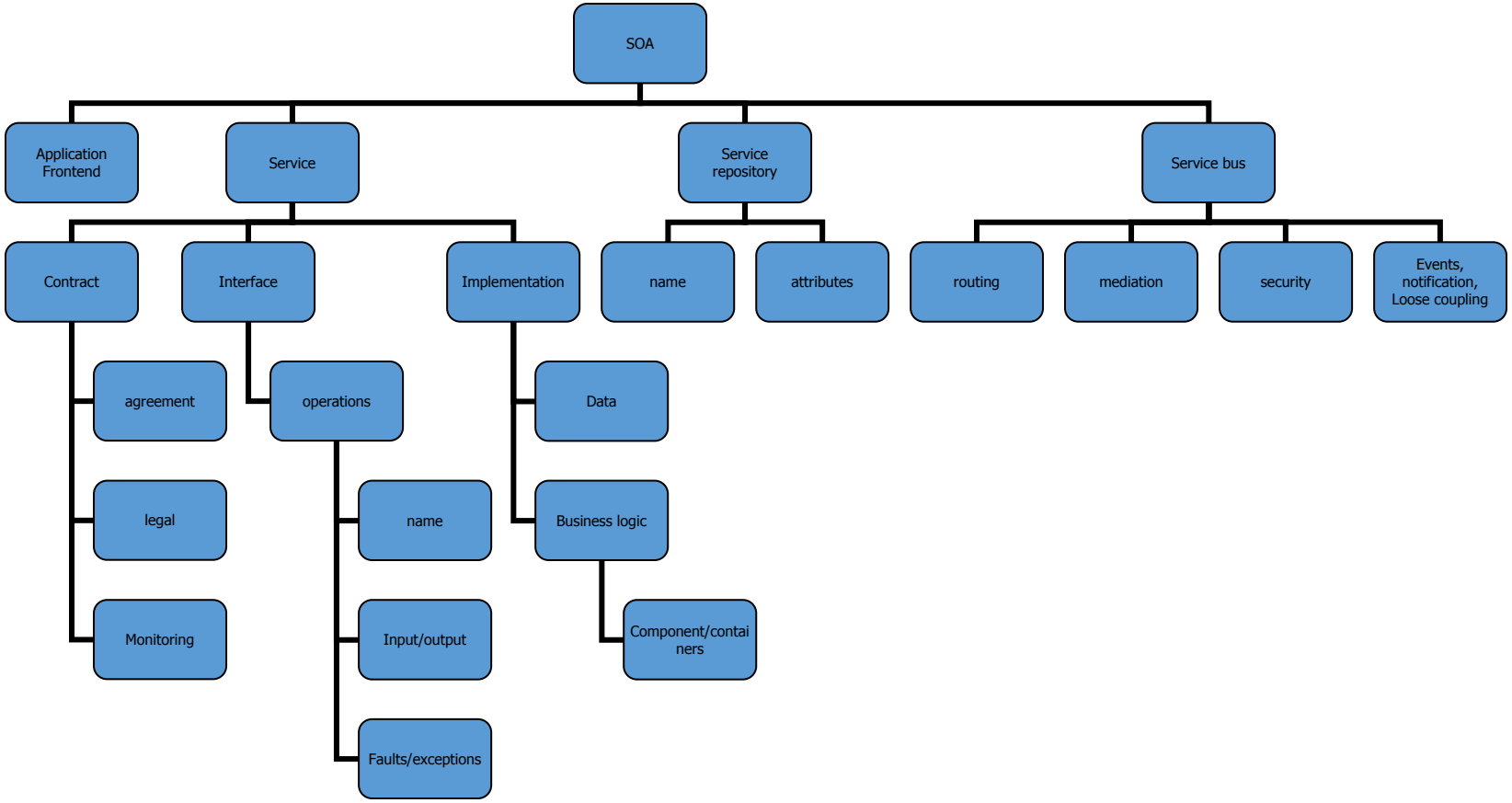
Elements of SOA



Elements of SOA

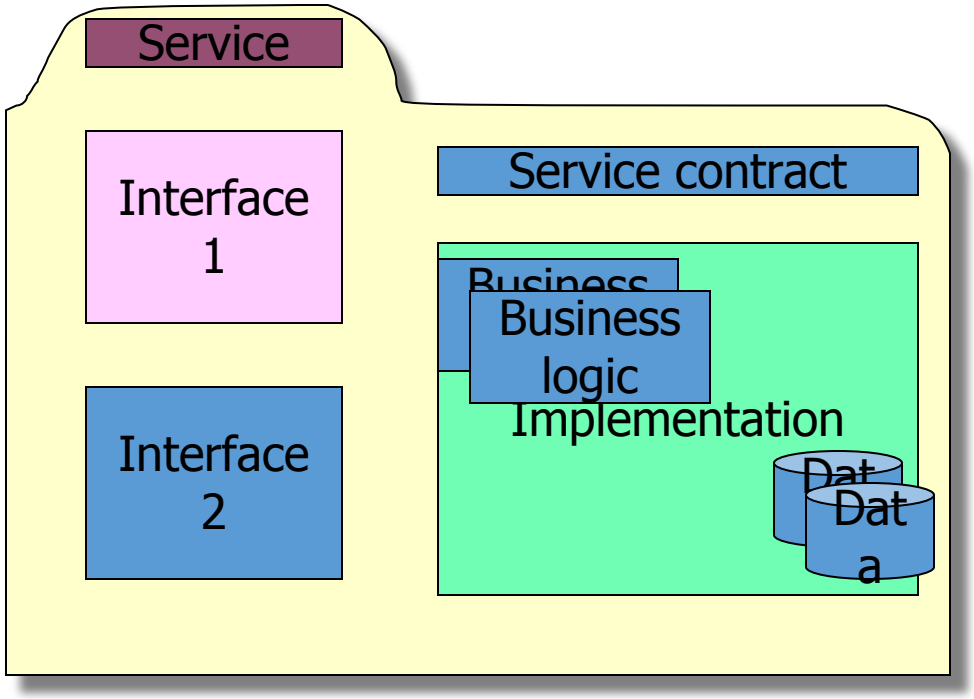
1. Application frontends: are active elements of the SOA, delivering the value of SOA to the end users.
   - They initiate and control all activity of the enterprise system.
   - Web application, application with GUI, or a batch application.
2. Service: a software component that encapsulates a high level business concept.
3. Contract: provides a specification of the purpose, functionality, constraints, and usage of services.
4. Interface: functionality of the service exposed by the service to the clients that are connected to the service.
5. Implementation: the service implementation provides the required business logic and appropriate data. It contains one or more of the artifacts: programs, configuration, data and databases.
6. Business logic: business process represented by the service.
7. Data: data represented in the service/ used by the service.
8. Service repository: it registers the services and their attributes to facilitate the discovery of services; operation, access rights, owner, qualities, etc.
9. (Enterprise) Service Bus (ESB): A flexible infrastructure for integrating applications and services by : routing messages, transforming protocols between requestor and service, handling business events and delivering them, providing QoS, mediation and security, and managing the interaction among services.
10. Open standards: publicly available implementable standards

Our view of SOA



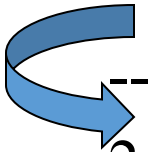
*

Service and Service Types



Types of services

1. Application frontend: GUI
2. Basic services: data and logic
3. Intermediary services: gateway, adapters
4. Process centric services: business operations
5. Public enterprise services: cross-enterprise: decoupling, security, governance



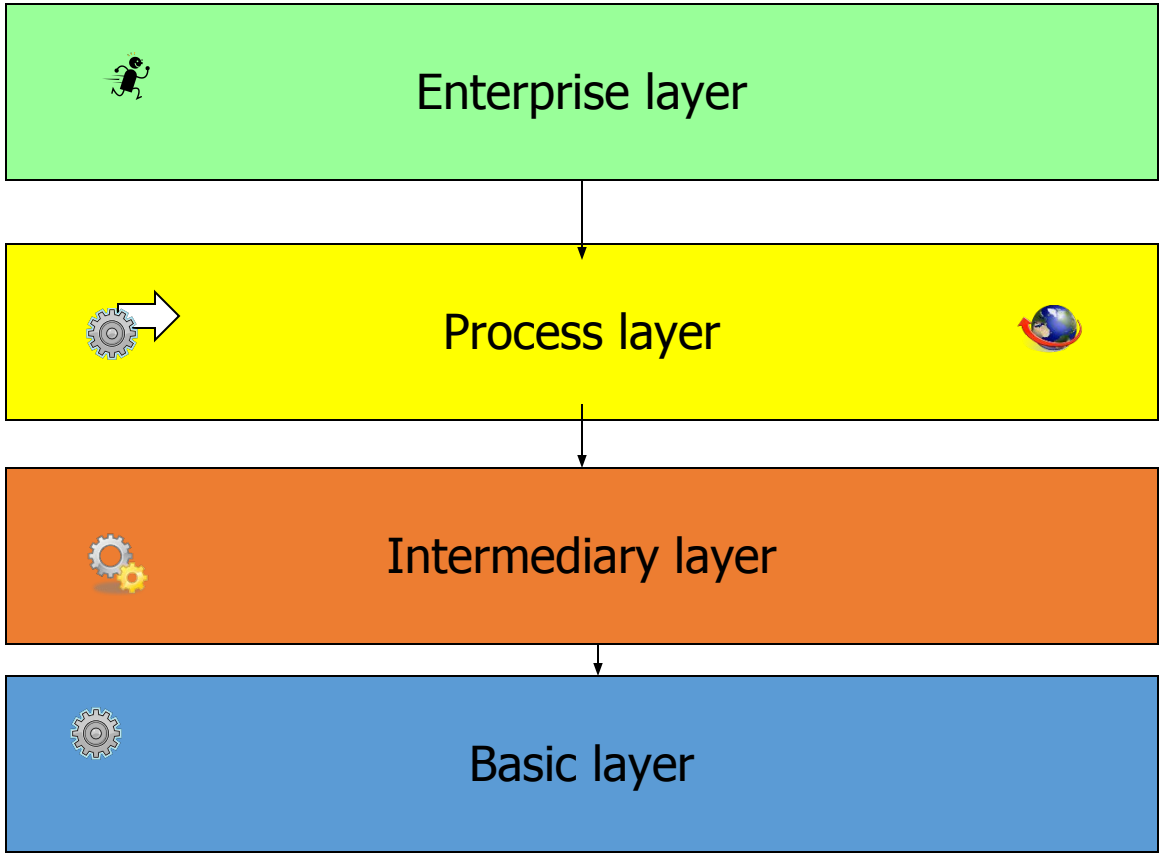
---

2 : horizontal services

4 : vertical services (domain-specific)

3 + 5: realized using ESB?

Enterprise Services layers



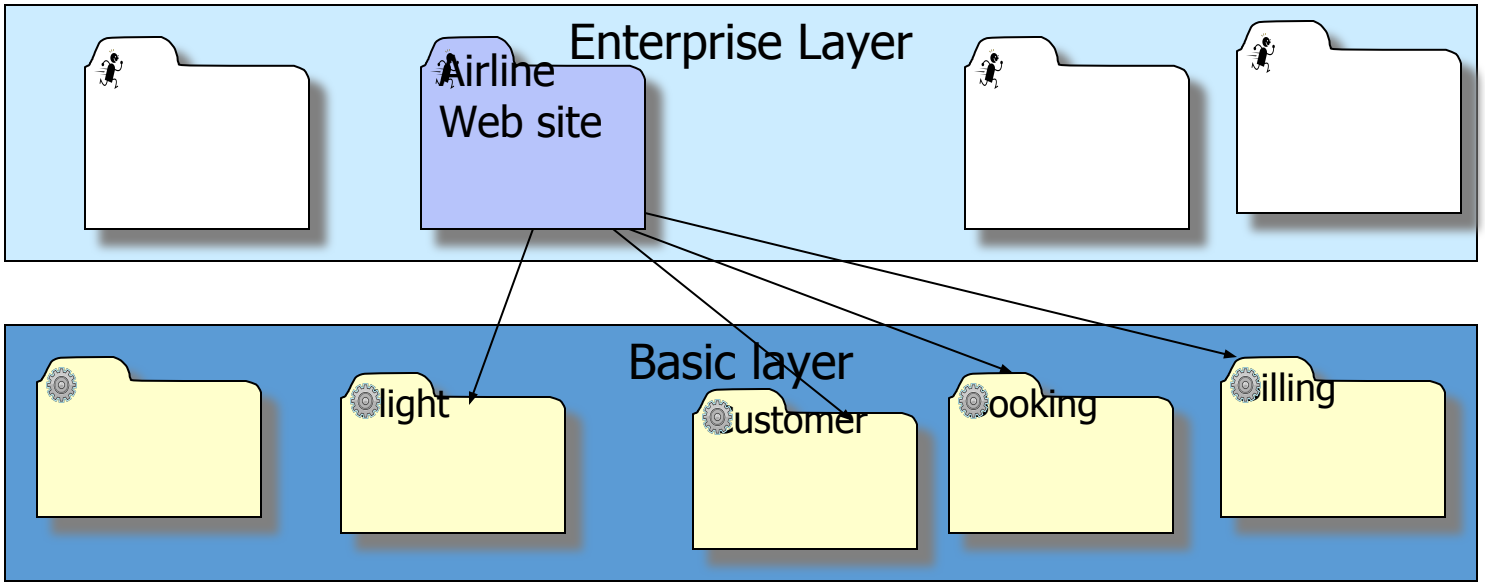
The architectural roadmap: The stages in development

- Fundamental SOA
  - Design fundamental services
- Networked SOA
  - Add intermediary services
- Process-enabled SOA
  - Add process-centric services, front-ends
- These three stages reach maturity at different rates, services gain more responsibility as the system matures.
- Advantages of using service-orientation will be apparent as the stages evolve independently of each other

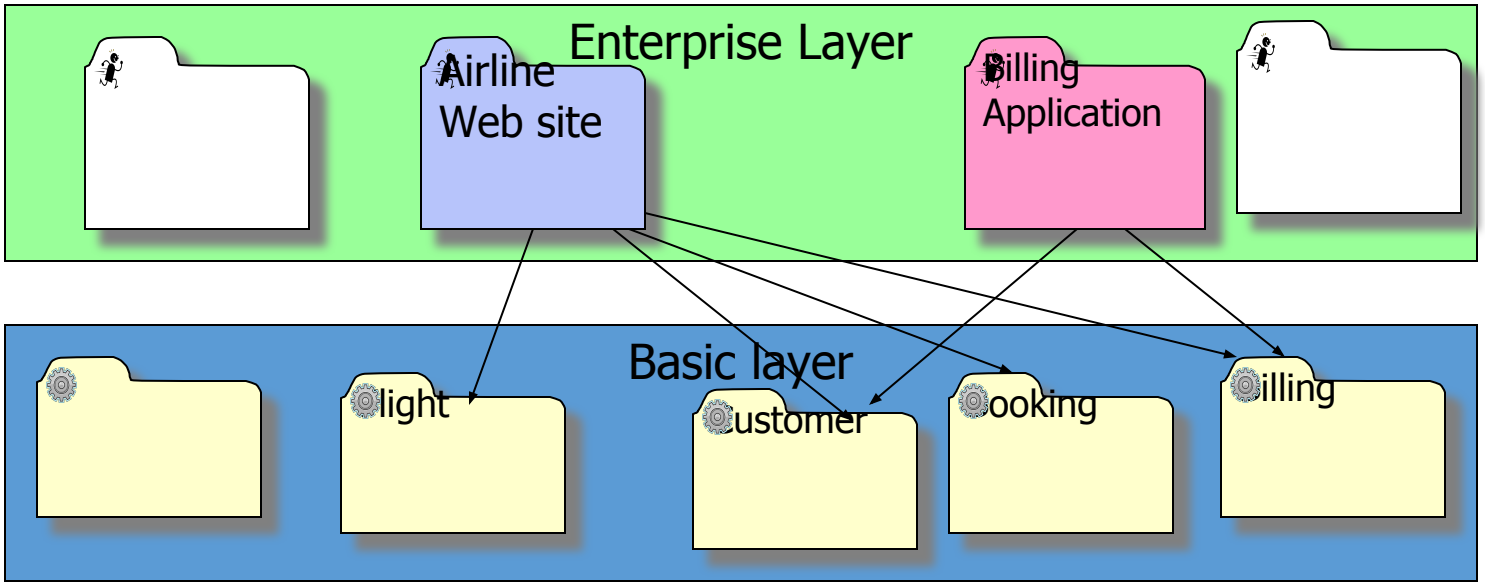
Fundamental SOA

- Excellent starting point for introduction SOA in an organization
- A fundamental SOA consists of two layers:
  - Enterprise layer that consists of front-ends, and
  - The basic layers that consists of basic services

Airline Enterprise



Expanded Airline Enterprise



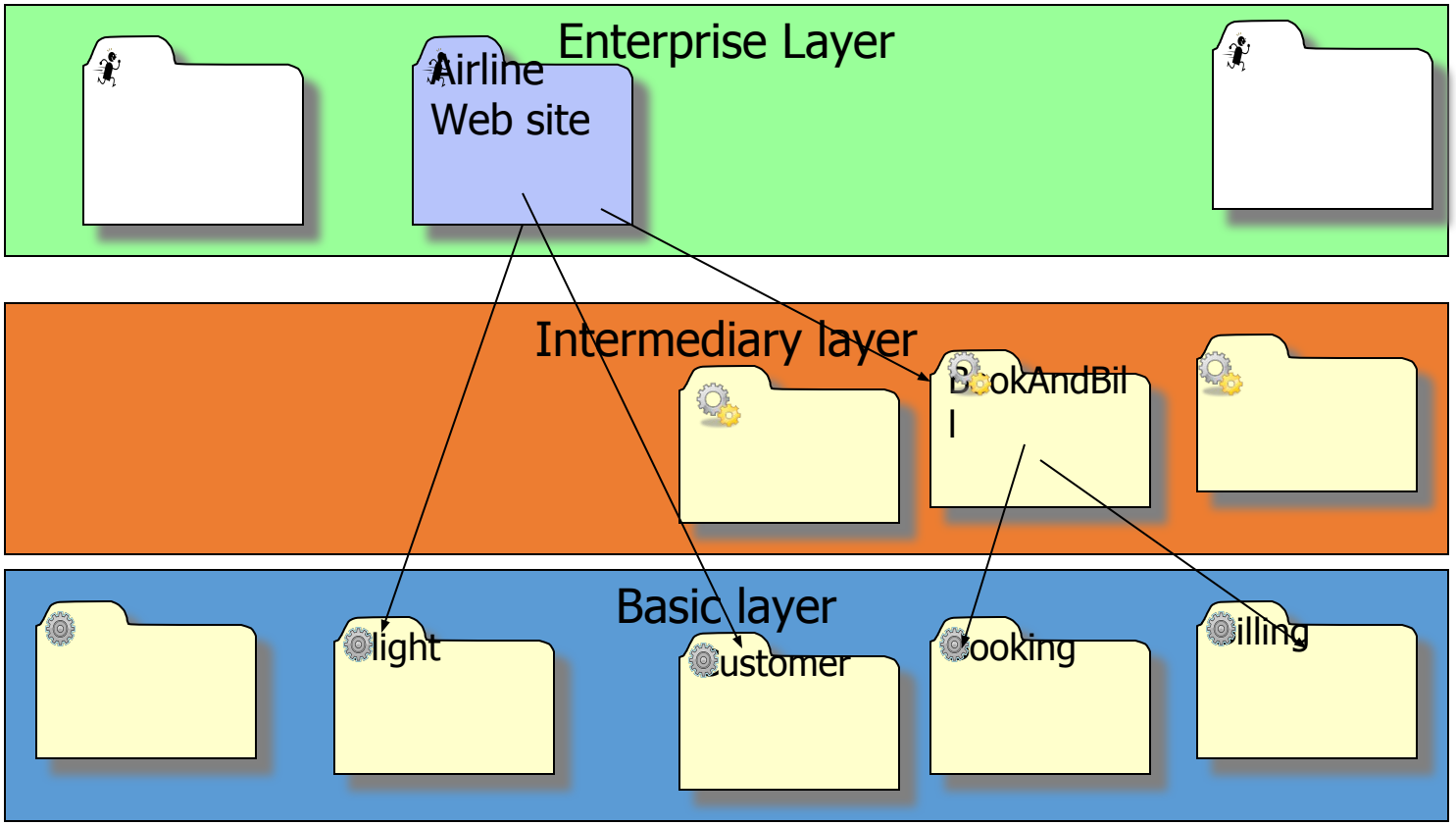
Fundamental SOA: Summary

- Base on which future expansion can take place
- Simple to implement
- Complex front-end
- Increased maintainability
- Shared services can make data replication largely obsolete
- Good starting point/entry point to SOA

Networked SOA

- It deals with backend complexity in addition to technical and conceptual integration.
- If offers flexibility in integrating software assets of an enterprise.
- Enables loose coupling
- Addition of intermediary layer with services that handle
  - distributed transactions,
  - bridge technology gaps,
  - database integration,
  - Add new functionality,
  - Wrap legacy applications/service

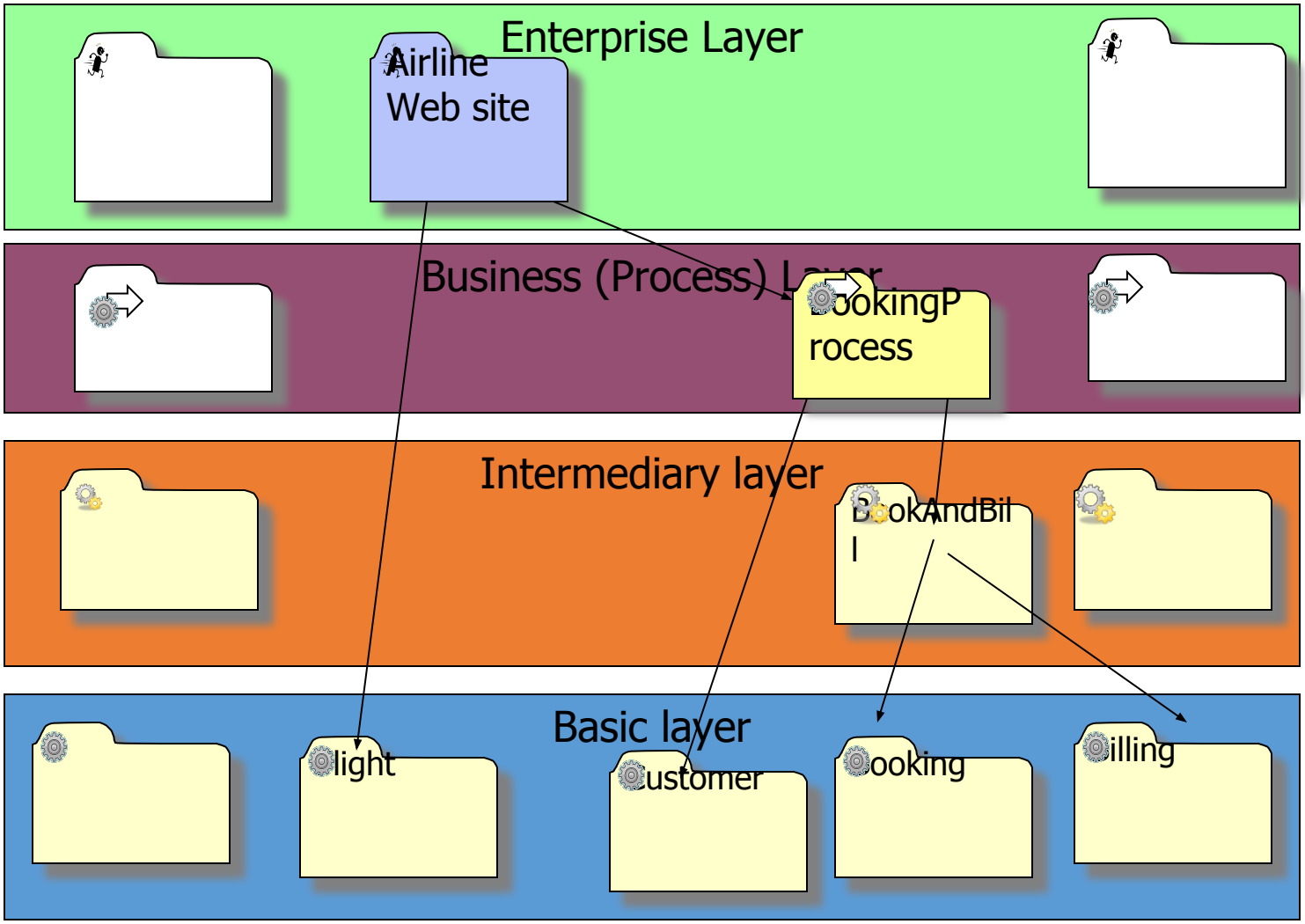
Networked SOA



Process-enabled SOA

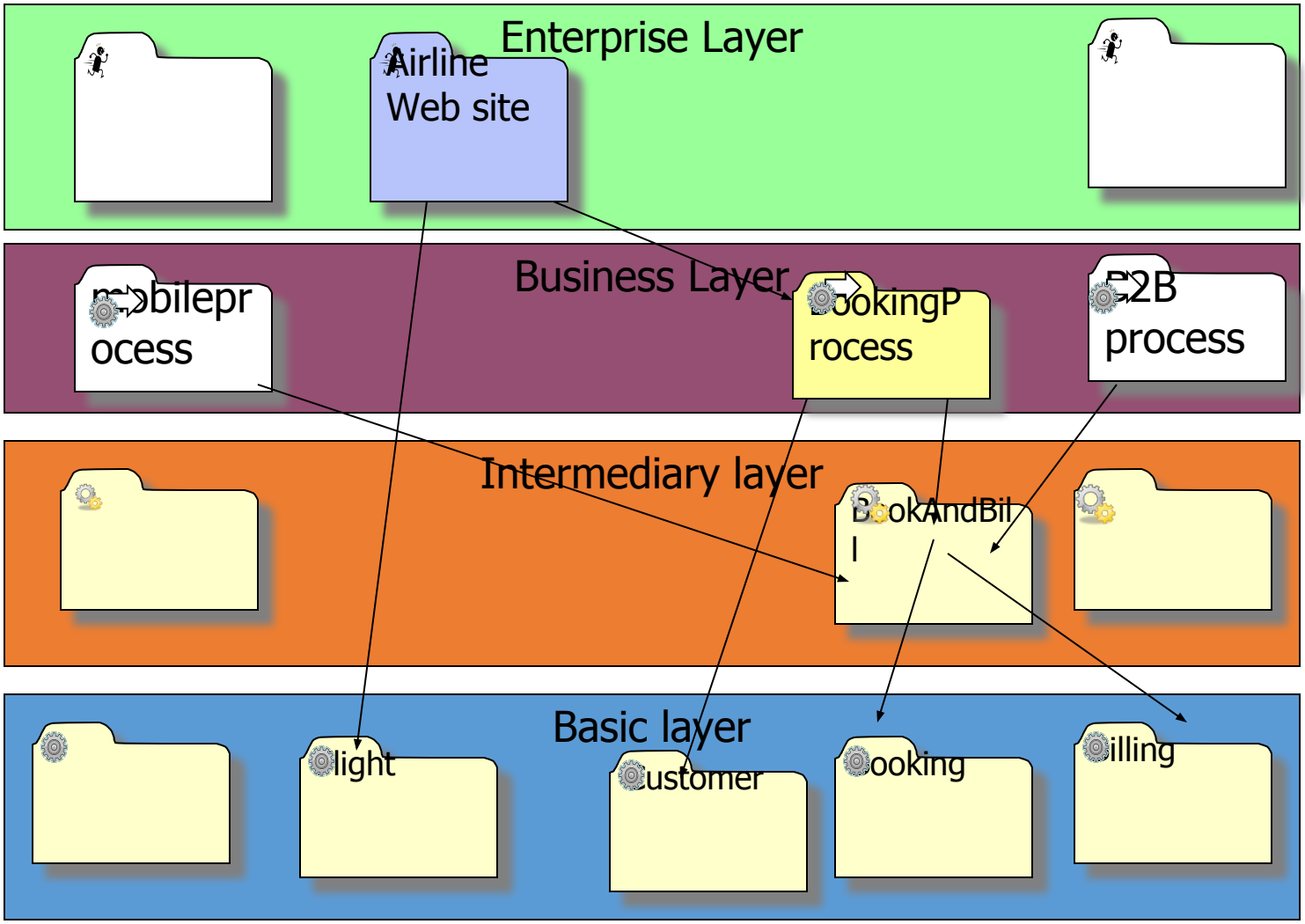
- The key feature is the maintenance of a process state in process-centric services.
- Stateful services (server-side state)
- Encapsulates complexity of processes (Ex: runExperiment in a complex scientific lab experiment)
- Possibility of sharing states between clients (Ex: research whiteboard)
- Handling long-living processes (Ex: auction framework)
- Enables the IT and business alignment

Process-enabled SOA

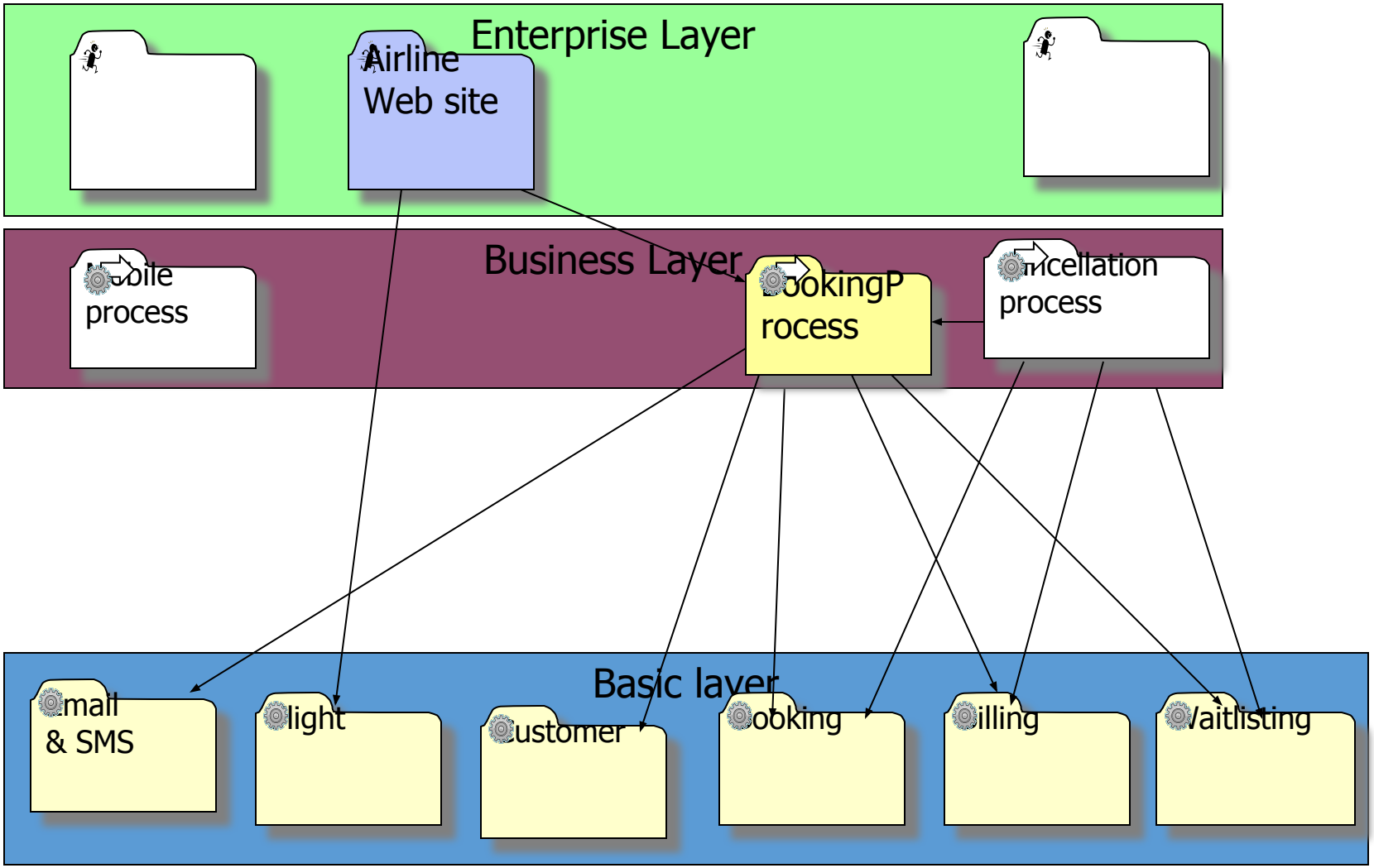


*

Process-enabled SOA (contd.)



Process-enabled SOA (contd.)

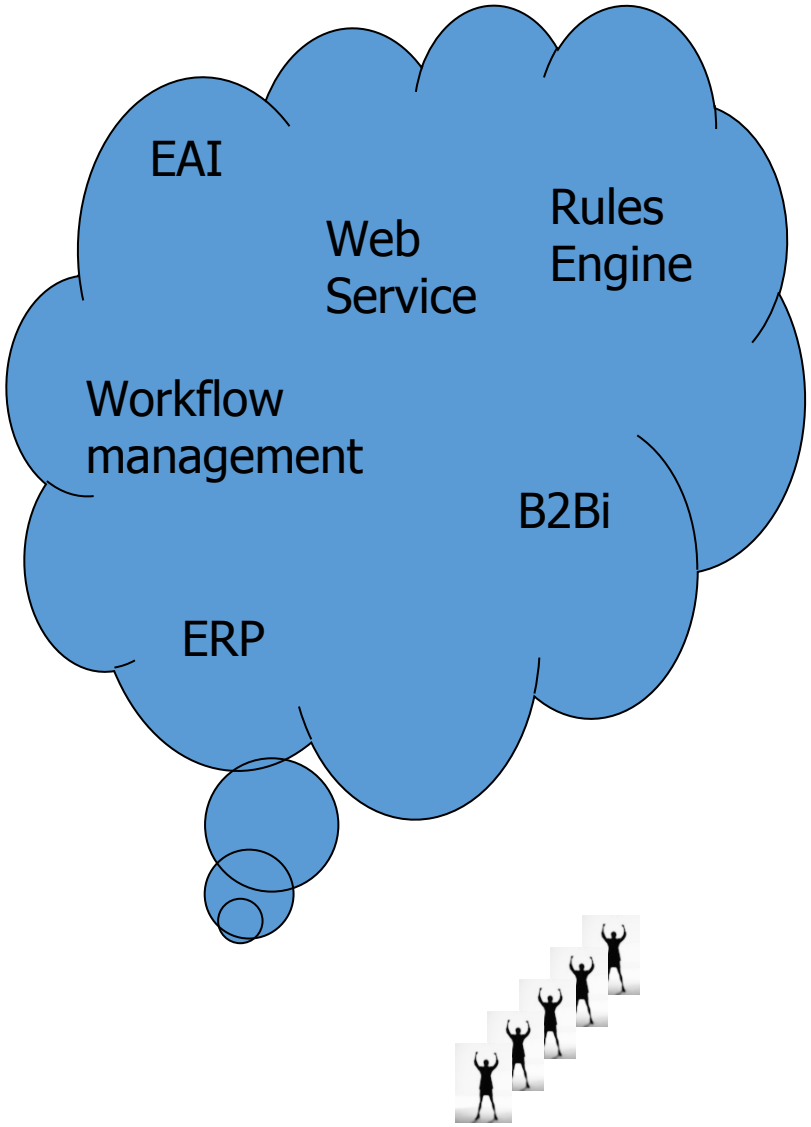


Process-enabled SOA summary

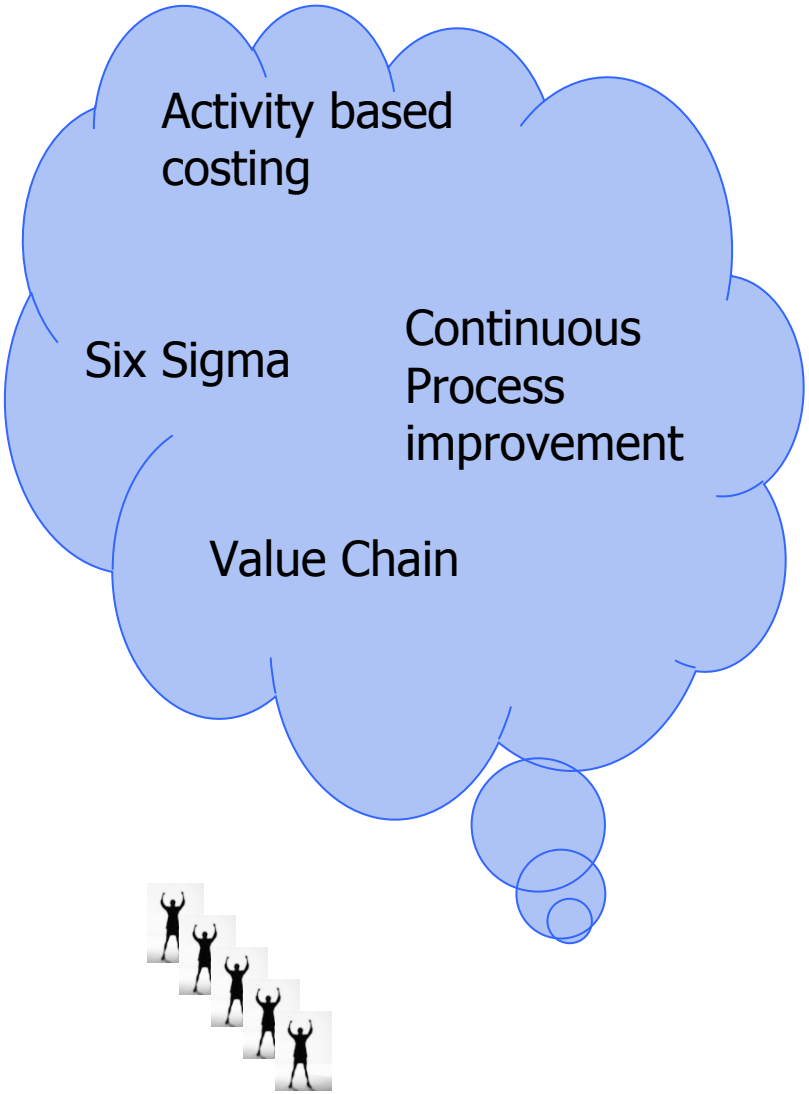
- Enables light-weight frontends (handles only user interaction)
- Encapsulates complexities of business processes
- Abstracts complexities of backend systems
- Enables separation of business logic from technology complexities
- Is required for integration of independent organizations and implementation of complex processes

Business Process Management (BPM)

- BPM generally focuses on the strategic and operational aspects of process orientation in a given business area.
- Mapping BPM model to an enterprise IT landscape is a challenging task.
  - Business side of BPM are the keywords such as ISO 9000 and Six Sigma
  - IT side of BPM is accompanied by keywords such a process modeling and workflow management



IT Organization



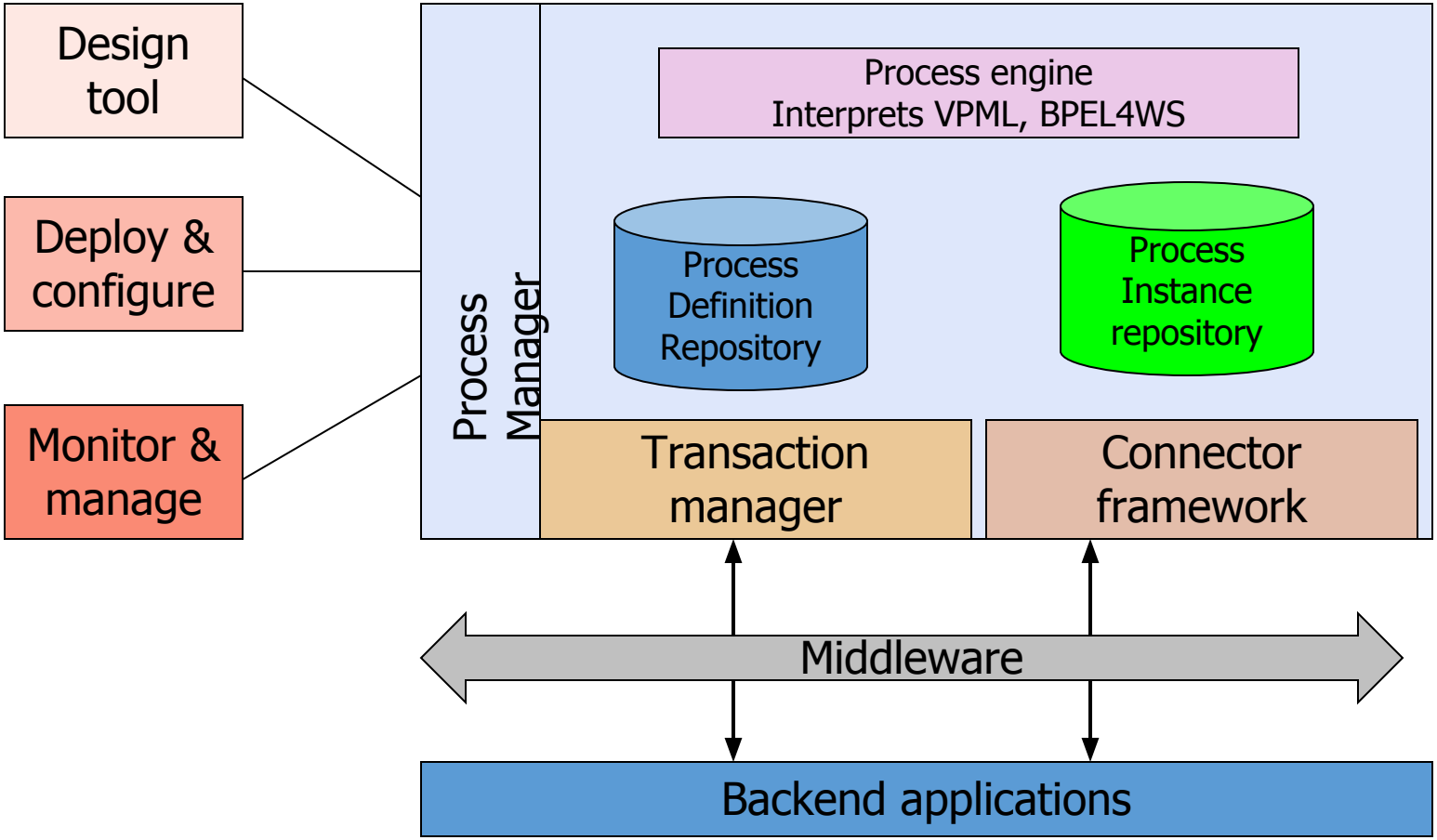
Business Organization

*

Business Process Management System (BPMS)

- BPMS provides the technical platform for realizing BPM management initiatives.
  - BPM engine, facilities for BPM monitoring, design tools, and facilities for simulation.
  - "BPM encompasses the discovery, design, and deployment of business processes, as well as executive, administrative and supervisory control over them to ensure that they remain compliant with business objectives" [SF03]
  - A BPM software product should enable business analysts, software developers, and system administrators to model and deploy business processes (at development time) and to interact with, monitor and analyze process instances 9at run time).
  - Lets discuss Modeling and execution architecture of BPMS.

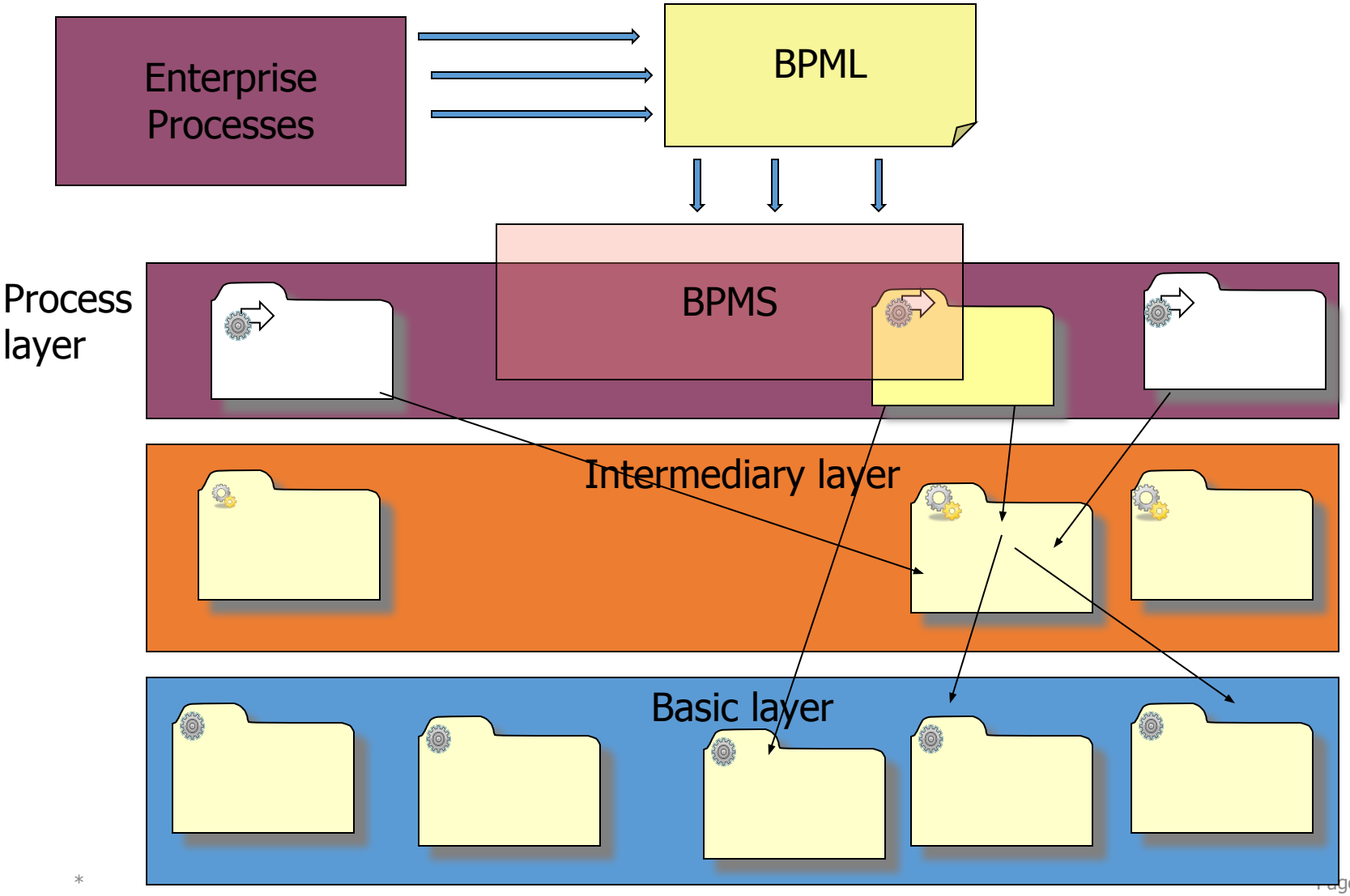
BPM System Architecture



BPM vision

- BPM vision is strong one
  - Instead of hard coding business processes into applications, it facilitates modeling, modifications, reconfigurations, and optimization of process definitions with graphical tools that can be used by less technology-oriented business analysts.

BPM Alignment to SOA



Web Services

- *Web Services* is a technology that allows for applications to communicate with each other in a standard format.
- A *Web Service* exposes an interface that can be accessed through messaging.
- Deployable unit.
- A Web service uses protocol to describe an operation and the data exchange with another web service. Ex: SOAP
- Platform independent, say, through WSDL.
- Publishable, discoverable, searchable, queryable
- Scalability issues: A group of web services collaborating accomplish the tasks of a large-scale application.
- Web services can be used to realize the "services" in an SOA.
- Your task in the first week is to review WS concepts,
- Try a simple implementation of a WS and get familiarized with WS framework (XML, SOAP, REST, WSDL etc.), if you have not done so.

Amazon.com and SOA

- "SOA creates order out of chaos @ Amazon" by Rich Seely (June 23, 2006) based on Werner Vogels' talk "Order in the Chaos: Building the Amazon.com Platform."
- 1995: Started out with a single web service on a single server. Today amazon has about 150 web services on its homepage alone.
- 1 million merchant partners; 60 million customers
- One server of customers and inventory grew into two servers; more database servers were added as the business expanded
- 1999: A mistep during this exponential growth period was moving to mainframe from distributed server. Failed to meet scalability, reliability and performance; it was scratched in 2000.

Amazon (contd.)

- Robustness: Shopping cart is tested for 20000 items by a single customer, for example!
- Amazon's secret sauce is "operating relaibly at scale".
- After "the denial of service" debacle in 1999, they decided to use Web services to insulate the databases from being overwhelmed by direct interaction with online applications.
- Each web service is the responsibility of a team of developers:
  - "And they are not just responsible for writing the service and then tossing it over the wall for testing and eventual entry into production where some poor maintenance geek has to look after it.
  - The Amazon CTO tells his Web services team members: "You build it. You own it."
  - That means the team is responsible for its Web service's on-going operation. If a Web service stops working in the middle of the night, team members are called to fix it."
- Web services are kept simple: complexity is the notorious enemy of reliability
- No attachment to one technology or standard: what ever customer wants, give it. (Ex: REST and SOAP)