SOE Notes

Primarily from Understanding SOA with Web Services[1]

SOE (Service-Oriented Enterprise) components:

- XML (Extensible Markup Language) (W3C): A common, independent data format across the enterprise and beyond that provides: (a) standard types and structures (.dtd, .xsd), independent of any programming language, development environment, or software system; (b) pervasive technology for defining business documents[2] and exchanging business information, including standard vocabularies for many industries; (c) ubiquitous software for handling operations on XML, including parsers, queries and transformations.
- WS (Web Services): XML-based technologies for messaging, service description, discovery, and extended features, providing: (a) pervasive, open standards for distributed computing interface descriptions and document exchange via messages; (b) independence from the underlying execution technology and application platforms; (c) extensibility for enterprise qualities of service such as security, reliability, and transactions; (d) support for composite applications such as business process flows, multi-channel access, and rapid integration.
- SOA (Services-Oriented Architecture): A methodology for achieving application interoperability and reuse of IT assets that features: (a) a strong architectural focus, including governance, processes, modeling and tools; (b) an ideal level of abstraction for aligning business needs and technical capabilities, and creating reusable, coarse-grained business functionality; (c) a deployment infrastructure on which new applications can quickly and easily be built; (d) a reusable library of services for common business and IT functions. Also, a methodology for building IT systems in which business services (requirements, customers, clients, processes…) are the key organizing principals. Resulting IT systems are ultimately tied to any underlying development environment technology ((CICS, IMS – mainframe), CORBA (OMG), J2EE, and COM/DCOM); may or may not be through WSDL (W3C) and SOAP (W3C) (object, procedure and data oriented development tend to tie the resulting systems directly to the underlying development environment technology).
- BPM (Business Process Management): Methodologies and technologies for automating business operations that: (a) explicitly describe business processes so that they are easier to understand, refine and optimize; (b) make it easier to quickly modify business processes as business requirements change; (c) automate previously manual business processes and enforce business rules; (d) provide real-time information and analysis on business processes for decision makers.

[1] Newcomer, Eric; Greg Lomow. Understanding SOA with Web Services; Addison Wesley, December 2004. ISBN: 0-321-18086-0.

[2] Sometimes an organization will know the key business documents that need to be exchanged (e.g. purchase order or employee record) prior to defining the services and the service interfaces, and other times, an organization will define the business documents and the service interface together. It is important that all related services share the same service-level data model, including the structure and semantics of the business documents.

SOE Notes

Design, implementation and management characteristics of an SOE (pp. 73-86):

- Primary:
  - Loosely coupled:
  - Well-formed service contracts:
  - Meaningful to service requesters:
  - Standards-based:
- Secondary:
  - Predictable service-level agreements:
  - Dynamic, discoverable, metadata-driven:
  - Design service contracts with related services in mind:
  - Implementation independent of other services:
  - Consider the need for compensating transactions:
  - Design for multiple invocation styles:
  - Stateless:
  - Design services with performance in mind:
- Requestors:
  - Service proxy types:
    - (1) Requestor creates entire SOAP message (envelope and body) (e.g. using XML-DOM) and call the transport-level API to send the SOAP message to the service endpoint.
    - (2) Requestor calls a document-oriented service proxy, passes the appropriate XML documents (that correspond to the WSDL parts for that operation) as parameters and the proxy validates the XML documents against data types (.dtd, .xsd), creates a SOAP message, calls transport-level API to send.
    - (3) Requestor calls an object-oriented service proxy (e.g. Java). The service proxy object has a method for each service operation. The parameters to the methods are objects generated according to the WSDL parts for that operation. When the service requestor calls the proxy, it passes the appropriate objects as parameters and the proxy converts the objects to XML documents, creates a SOAP message by building the SOAP envelope and using the XML documents to assemble the body of the SOAP message, and then calls the transport-level API.
- Legacy systems and services:
  - Service contracts:
    - Directly to applications:
    - To middleware:

General SOE development environments:

- Data oriented development:
- Procedure oriented development:
- Object oriented development: e.g. CORBA development environment; common programming interface (IDL), common interoperability protocol (IIOP). Object defined by method signatures.
- Service oriented development: e.g. SOE development environment based on Web Services; common programming interface (WSDL), common interoperoperability protocol (SOAP); suited for satisfying interoperability across object, procedure, data environments (introduces another level

of indirection). A service is defined by messages exchanged with other services, and at a higher abstraction level (lowest common denominator) than object-oriented (might have to interface to a procedural language (e.g. COBOL, PL/I), a queuing system (e.g. JMS, MSMQ), and/or an object-oriented system (e.g. J2EE, .NET)).

Agent oriented development: e.g. JADE development environment; emphasizes the fundamental role of actors/agents, and of communication and interaction for analyzing and designing (organizations and organizational (e.g. business modeling)) information systems.

SOA implementation technologies (pp. 150,159):

Distributed objects: CORBA, J2EE, COM/DCOM.
Message-oriented middleware (MOM): WebSphere MQ, Tibco Rendezvous.
Transaction Processing (TP) monitors: (CICS, IMS – mainframe), Encinia, Tuxedo.
Homegrown middleware: in-house.
B2B platforms: ebXML, RosettaNet.

SOA technical benefits (pp. 86-92):

Efficient development:
More reuse:
Simplified maintenance:
Incremental adoption:
Graceful evolution:

SOA business benefits (pp. 93-102):

Increased business agility:
Velocity: reducing the amount of time required to assemble new business applications from existing services and IT assets.
Flexibility: easier and less expensive to reconfigure and adapt services and IT assets to meet new (unanticipated) functional/non-functional requirements.
Better business alignment with technology: encourages modeling business processes and a direct mapping to IT services.
Improved customer satisfaction:
Reduced vendor lock-in and reduced switching costs:
Application platform (J2EE, .NET Framework, Oracle, (CICS, IMS – mainframe))
Packaged applications (SAP, PeopleSoft, Siebel)
Middleware technology (WebSphere MQ, Tibco, CORBA)
Specific product features (stored procedures, cluster cashing)
Reduced integration costs: because SOA is based on loosely coupled services with well-defined, platform-neutral service contracts.
Improved ROI of existing IT assets: by reusing existing assets as SOA services.

SOE Notes

XML and Web Services, integration and interoperability implementation approaches (pp. 171-178):

- WSI (Web Services Integration): Tactical, opportunistic; works best when immediate results and short-term ROI are required and when longer-term costs are less important (a series of point-to-point integrations).
- SOI (Service-oriented Integration): Strategic, systematic; works best for organizations that are trying to maximize long-term results of an integration architecture by heavily investing in one.

Challenges to adoption:

- The definition of a reusable service is difficult to get right the first time.
- Re-engineering existing systems costs money; payback takes time.
- Business analysts must define business processes; systems architects must turn processes into specifications; software engineers must develop new code; project managers must track progress.
- Some applications may not have callable interfaces that can be service enabled (e.g. only file accessible, or batch input/out accessible).
- Identify/begin those projects that provide immediate value, while laying a foundation for future departmental/enterprise SOA development.

Execution environment patterns for exchanging messages between applications

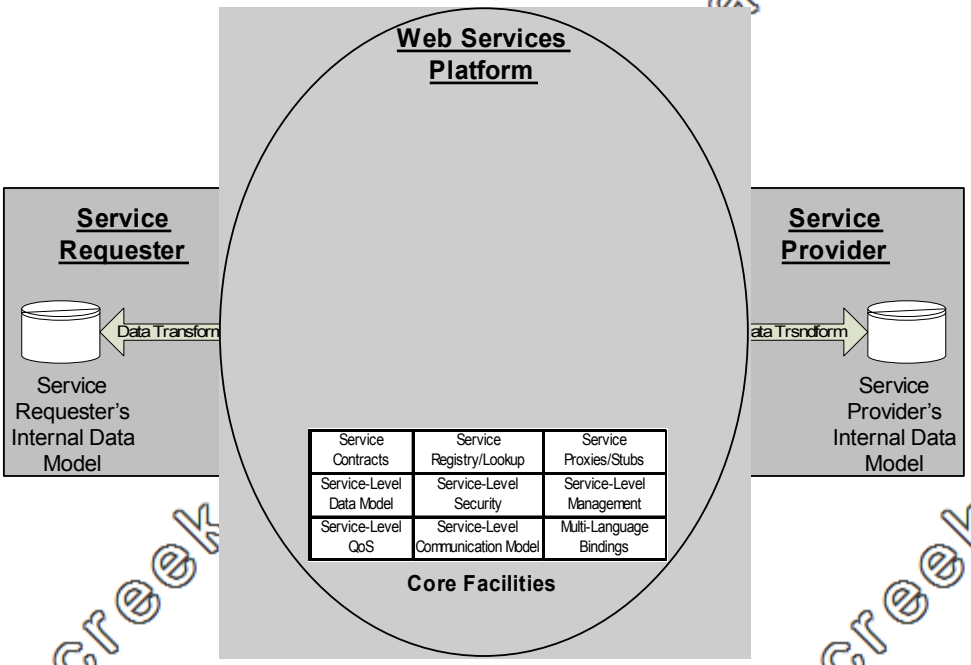
- Browser/HTML (W3C)/HTTP (W3C):
- Request/response:
- Request/callback:
- Asynchronous store-and-forward messaging:
- Publish/subscribe:

WebServices

- Core Facilities (pp 103-159):
  - Service Contracts: a well defined, formal interface to a service. It clearly defines what the service does, and separates the requester from the service's technical implementation. Both the requestor and the provider need to be able to understand the (WSDL) service contract. WSDL contracts separate the logical portion (port types, operations, messages, parts, data types) from the physical portion (transport bindings (e.g. HTTPS), programming languages (e.g. Java), wire-level data formats (e.g. XML)). The service contract defines a data model.
    - Elements:
      - Service names:
      - Version number:
      - Pre-conditions:
      - Service classification:
    - For each operation:
      - Operation name:
      - Pre-conditions:
      - Post-conditions:
      - Input data profile:
      - Output data profile:
      - Interaction profile:
      - Exception conditions and error handling:

Security profile:
Transactional profile and recovery semantics:
Service-level management agreement:

Service Registry/Lookup:
Service Proxies/Stubs:
Service-Level Data Model (XLM schema): made up of all the XML data types of all the input documents/messages and output documents/messages of the operations that make up the service (which are defined in the WSDL service contract). This data model is independent of the data types that the service requesters and service providers use internally. Requester and provider may have to perform data transformations (e.g. using XML Schema (.xsd), XSLT) to translate data elements to/from the service-level data model.



Service-Level Security:
Service-Level Management:
Service-Level QoS:
Service-Level Communication Model:
Multi-Language Bindings:

Extended Facilities:

Metadata Management: for defining ways in which cooperating Web services can discover the features other services support. May be dynamic.
XML Schema: message data typing and structuring, and expressing policy information.
WSDL: associates messages and message exchange patterns with service names and network addresses.
WS-Addressing (W3C): replaces WS-Routing, WS-Referral; includes endpoint addressing and reference properties associated with endpoints.
WS-MessageDelivery:
WS-Policy (W3C): associates quality of service requirements with a WSDL definition.

SOE Notes

Web Services Policy Language (OASIS):
WS-MetadataExchange: queries and discovers metadata associated with a Web service. Essentially replaces UDDI for most applications.

Security: for ensuring privacy (encryption), message integrity, authorization, and authorized access (authentication); applied to the network layer, message layer and data layer.
WS-Security (OASIS): provides message-level security on an end-to-end basis for Web services messages.
WS-SecurityPolicy: defines security assertions detailing a Web service's requirements so that the service requestor can meet them.
WS-Trust: defines how to establish overall trust of the security system by acquiring any needed security tokens (e.g. Kerberos tickets) from trusted sources.
WS-SecureConversation: defines how to establish and maintain a persistent context for a secure session over which multiple Web service invocations might be sent without requiring expensive authentication each time.
WS-Federation: defines how to bridge multiple security domains into a federated session so that a Web service only has to be authenticated once to access Web services deployed in multiple security domains.
XML Signature (W3C): designed to provide integrity, using a variety of encryption and signature mechanisms, to ensure that service providers can determine whether or not documents have been altered in transit ant that they are received once and only once.
XML Encryption (W3C): designed to provide confidentiality, using a variety of encryption algorithms, of part or all of an XML document to ensure that the contents of a document cannot be intercepted and read by unauthorized persons.

Reliability: for ensuring that SOAP messages are delivered (over potentially unreliable networks such as HTTP) and processed.
WS-Reliability (OASIS):
WS-ReliableMessaging (IBM, Microsoft):

Notification: for defining additional message exchange patterns such as broadcast and publish/subscribe.
WS-Eventing:
WS-Notification (OASIS):

Transactions: for coordinating the work of multiple independent Web services into a larger unit.
WS-Transactions family (BEA, IBM, Microsoft):
WS-AtomicTransactions: defines volatile and durable variations of a standard two-phase commit protocol for short-lived executions.
WS-BusinessActivity: defines variations on the idea of tentative commit and compensation-based undo protocols for longer-lived executions.
WS-Coordination: defines the coordinator for the above two pluggable protocols (and their variations).
WS-Composite Application Framework (WS-CAF) (OASIS): breaks context management into a separate specification and adds a third transaction protocol specifically designed for business process management.
WS-Context: defines a standalone context management system for generic context (e.g. for non-transaction protocol contexts such as security, device and network IDs, or database and file IDs).

WS-CoordinationFramework: defines a coordinator for the basic context specification and the pluggable transaction protocols in the WS-TransactionManagement specification.

WS-TransactionManagement: defines three transaction protocols for the pluggable coordinator: ACID, long-running actions (compensation) and business process management.

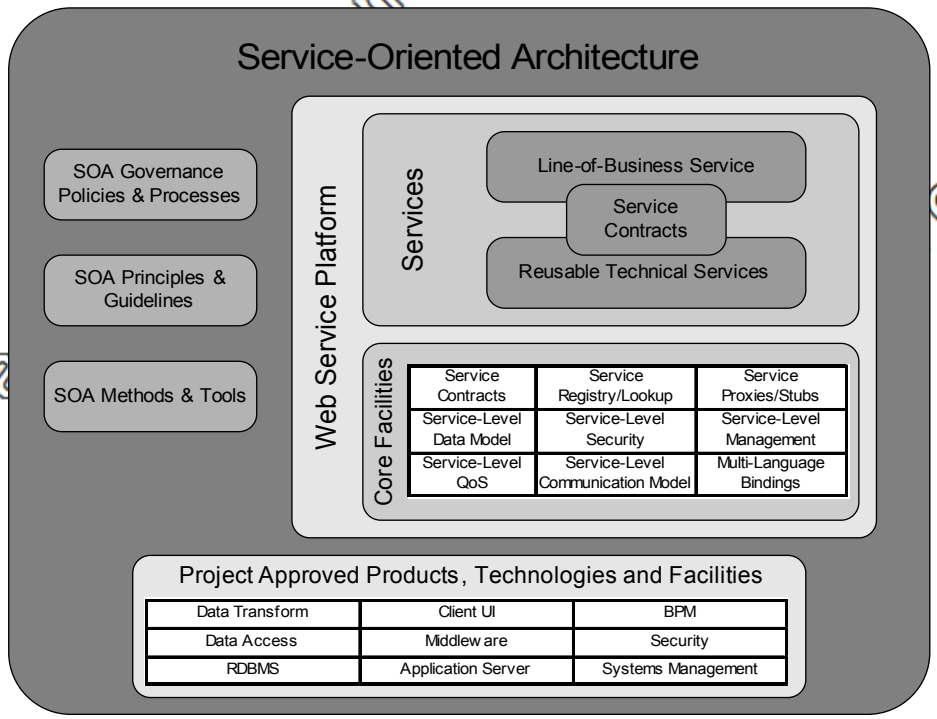
Orchestration: for combining multiple Web services to perform a larger unit of work. An orchestration model is between two participants (specifically focusing on the view of one participant), as opposed to a choreography model which provides a larger scope, encompassing all parties and their associated interactions (e.g. a peer to peer model).

WS-BPEL (OASIS): assumes that Web services are defined using WSDL and policy assertions that identify and extended features. Typically, a flow is initiated by the arrival of an XML document, and so the document-oriented (driven) Web services style (as opposed to model-, quality-, data-, schema-, ontology-, constraint-) tends to be used for modeling the entry point to a flow. Parts of the documents are typically extracted and operated upon by the individual tasks in the flow, such as checking on the inventory availability for each line item from a different supplier, meaning the steps in the flow may be implemented using a combination of request/response and document-oriented Web services. The WS-BPEL specification differs from other extended specifications in that it defines an executable language compatible with various software systems that drive business process automation. Whereas most other Web services specifications are XML representations of existing distributed computing features and capabilities that extend SOAP headers, orchestration represents the requirement for composing Web services in a declarative (non-programmatic) manner. Using Web services orchestration for BPM is clearly more difficult because it requires a deep understanding of an enterprise's business processes. In any case, it is true that the orchestration layer is where everyone expects the solution to be found to the hard problems of data type and structure incompatibilities, semantic data matching, and correlating the results of multiple Web services.

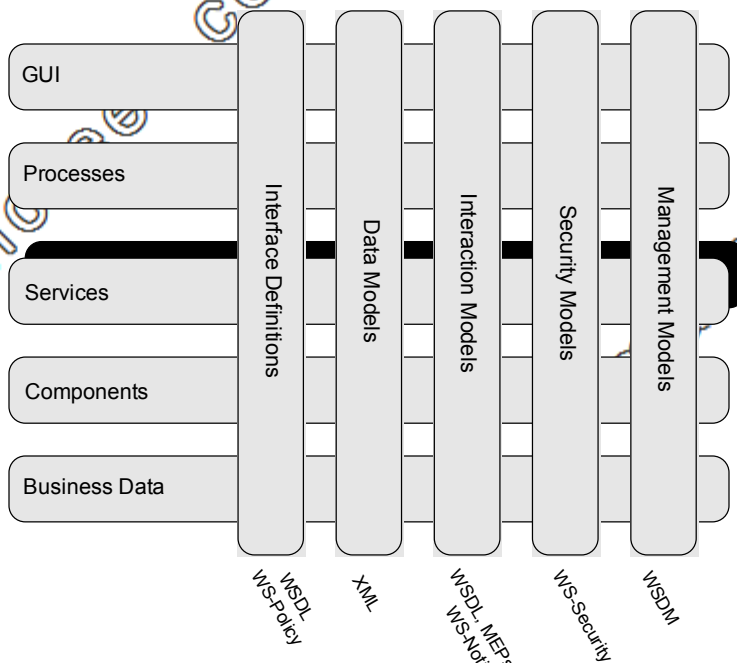
Web Services Choreography Definition Language (WS-CDL): establishes the formal relationship between two or more external trading partners. One difference between WS-CDL and WS-BPEL is that WS-CDL does not require Web services infrastructure at all of the endpoints being integrated.

SOE Notes

SOA key elements (pp. 54-70):



Services-level abstraction (pp.58-61):



SOE Notes

General Integration and Interoperability:

- Reconciling the differences between multiple IT systems whenever those systems need to interact.
- Business Drivers for Integration:
  - Mergers and acquisitions
  - Internal organization
  - Application/system consolidation
  - Inconsistent/duplicated/fragmentized data
  - New business strategies
  - Government regulation compliance
  - Business process streamlining
- Common challenges:
  - Reconcile incompatible business processes
  - Reconcile data differences
  - Reconcile incompatible technologies
  - Reconcile different time scales (e.g. sub-second vs. daily granularity)
  - Reconcile the different interaction patterns (e.g. synchronous vs. asynchronous)
- Typical requirements:
  - Inexpensive, fast ROI
  - Easy to learn / administer
  - Non-invasive
  - Scalable, reliable, high availability, fault tolerant, secure…
  - Easily adaptable
  - Some combination of strategic and tactical

Systematic	Opportunistic
Invest for future	Quick and dirty
Enterprise solution	Point solution
Fastest time-to-market across multiple projects	Fastest time-to-market for given product
Lowest cost across multiple projects	Lowest cost for given product
Enterprise data model	Ad hock data models for specific products
Enterprise integration backbone	Point-to-point integrations for specific product
Loosely coupled integration across projects / admin. boundaries	Tightly coupled integration, quick for given project
Standards-based, vendor/technology neutral	Quick, proprietary, vendor specific standards
Enterprise QoS incl. transactions and security	Product specific transactions and security

"Be strategic when you can, be tactical when you have to."

SOE Notes

Integration layer levels:

- Data (reconcile schemas and definitions)
- Message (translate message meanings between applications)
- Component (integrate CORBA, .NET, J2EE)
- Application (integrate at code level)
- Service (install / change at layer of indirection)
- Process (cross-organization agreement)
- User interface (at presentation / portal level)
- B2B (reconcile across multiple companies)

Web Services Integration and Interoperability:

Free form exchange of XML documents using HTTP, SOAP and a service registry.

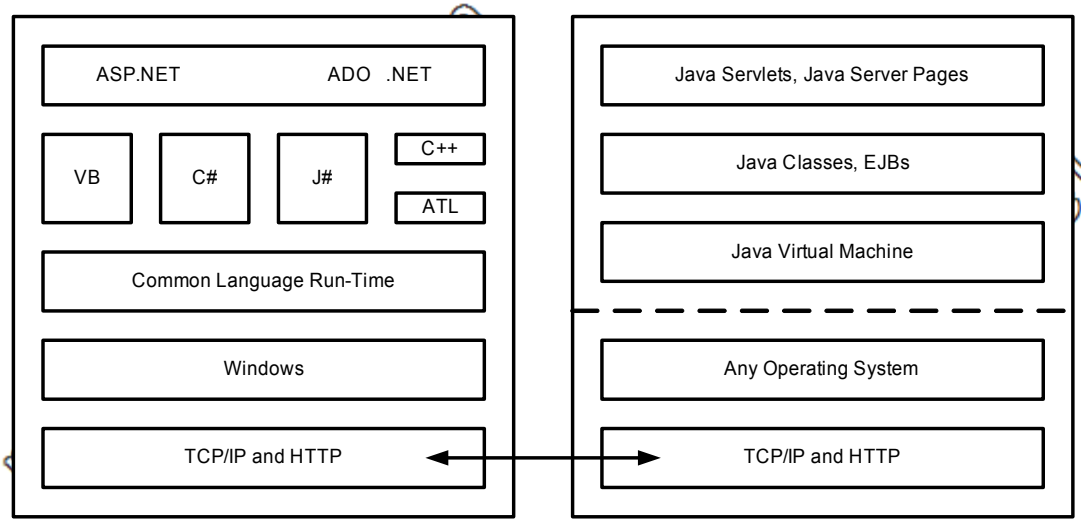
Common approaches:

- Legacy data-driven (develop XML schema, use SOAP as message format)
- API/method-driven (pick APIs/methods to be exposed, define XML data types, use SOAP)
- Contract-driven (define contracts based on bus. data, wrap systems to interpret, use SOAP)

Approaches:

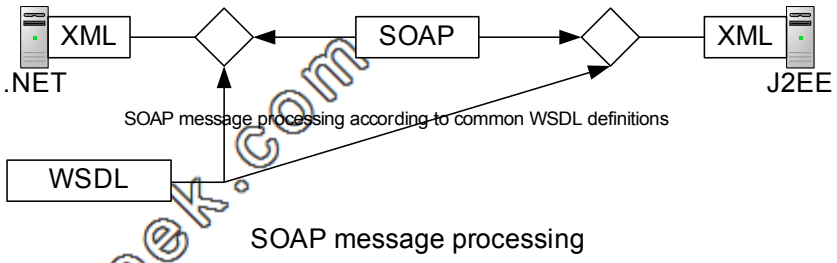
- Tactical (Web services integration (WSI))
- Strategic (Service-oriented integration (SOI))

J2EE and .NET Interoperability:



J2EE and .NET architectures

Limited to a fairly high level of abstraction. Define service contracts that exchange coarse-grained data objects or encapsulate multiple method invocations into a single WSDL service (e.g. define an XML schema (a.k.a. service-level data model) based on shared customer data, define appropriate WSDL operations based on the schema/data/needs, generate SOAP messages based on the environment).



SOAP message processing