SOA Design Patterns Reference Poster (Contributed by Jageshwar Tripathi, completesoa.blogspot.com)

Foundational Inventory

Enterprise Inventory (Erl)
How can services be delivered to maximize recomposition?

Domain Inventory (Erl)
How can services be delivered to maximize recomposition when enterprise-wide standardization is not possible?

Service Normalization (Erl)
How can a service inventory avoid redundant service logic?

Logic Centralization (Erl)
How can the misuse of redundant service logic be avoided?

Service Layers (Erl)
How can the services in an inventory be organized based on functional commonality?

Canonical Protocol (Erl)
How can services be designed to avoid protocol bridging?

Canonical Schema (Erl)
How can services be designed to avoid data model transformation?

Inventory Implementation

Dual Protocols (Erl)
How can a service inventory overcome the limitations of its canonical protocol while still remaining standardized?

Canonical Resources (Erl)
How can unnecessary infrastructure resource disparity be avoided?

State Repository (Erl)
How can service state data be persisted for extended periods without consuming service runtime resources?

Stateful Services (Erl)
How can service state data be persisted and managed without consuming service runtime resources?

Service Grid (Chappell)
How can deferred service state data be scaled and kept fault-tolerant?

Inventory Endpoint (Erl)
How can a service inventory be shielded from external access while still offering service capabilities to external consumers?

Cross-Domain Utility Layer (Erl)
How can redundant utility logic be avoided across domain service inventories?

Foundational Service

Functional Decomposition (Erl)
How can a large business problem be solved without having to build a standalone body of solution logic?

Service Encapsulation (Erl)
How can solution logic be made available as a resource of the enterprise?

Agnostic Context (Erl)
How can multipurpose service logic be positioned as an effective enterprise resource?

Non-Agnostic Context (Erl)
How can single-purpose service logic be positioned as an effective enterprise resource?

Agnostic Capability (Erl)
How can multipurpose service logic be made effectively consumable and composable?

Inventory Centralization

Process Centralization (Erl)
How can abstracted business process logic be centrally governed?

Schema Centralization (Erl)
How can service contracts be designed to avoid redundant data representation?

Policy Centralization (Erl)
How can policies be normalized and consistently enforced across multiple services?

Rules Centralization (Erl)
How can business rules be abstracted and centrally governed?

Logical Inventory Layer

Utility Abstraction (Erl)
How can common non-business centric logic be separated, reused, and independently governed?

Entity Abstraction (Erl)
How can agnostic business logic be separated, reused, and governed independently?

Process Abstraction (Erl)
How can non-agnostic process logic be separated and governed independently?

Inventory Governance

Canonical Expression (Erl)
How can service contracts be consistently understood and interpreted?

Metadata Centralization (Erl)
How can service metadata be centrally published and governed?

Canonical Versioning (Erl)
How can service contracts within the same service inventory be versioned with minimal impact?

Service Implementation

Service Facade (Erl)
How can a service accommodate changes to its contract or implementation while allowing the core service logic to evolve independently?

Redundant Implementation (Erl)
How can the reliability and availability of a service be increased?

Service Data Replication (Erl)
How can service autonomy be preserved when services require access to shared data sources?

Partial State Deferral (Erl)
How can services be designed to optimize resource consumption while still remaining stateful?

Partial Validation (Orchard, Riley)
How can unnecessary data validation be avoided?

UI Mediator (Utschig, Maier, Trops, Normann, Winterberg)
How can a service-oriented solution provide a consistent, interactive user experience?

Legacy Encapsulation

Legacy Wrapper (Erl, Roy)
How can wrapper services with non-standard contracts be prevented from spreading indirect consumer-to-implementation coupling?

Multi-Channel Endpoint (Roy)
How can legacy logic fragmented and duplicated for different delivery channels be centrally consolidated?

File Gateway (Roy)
How can service logic interact with legacy systems that can only share information by exchanging files?

Composition Implementation

Agnostic Sub-Controller (Erl)
How can agnostic, cross-entity composition logic be separated, reused, and governed independently?

Composition Autonomy (Erl)
How can compositions be implemented to minimize loss of autonomy?

Atomic Service Transaction (Erl)
How can a transaction with rollback capability be propagated across messaging-based services?

Compensating Service Transaction (Utschig, Maier, Trops, Normann, Winterberg, Loesgen, Little)
How can composition runtime exceptions be consistently accommodated without requiring services to lock resources?

Transformation

Data Model Transformation (Erl)
How can services interoperate when using different data models for the same type of data?

Data Format Transformation (Little, Rischbeck, Simon)
How can services interact with programs that communicate with different data formats?

Protocol Bridging (Little, Rischbeck, Simon)
How can a service exchange data with consumers that use different communication protocols?

This poster displays the catalog of patterns from the book SOA Design Patterns (by Thomas Erl, Prentice Hall, Copyright 2008).
For more information, visit www.soapatterns.org and www.soapatterns.com.

Service Security

Exception Shielding
(Hogg, Smith, Chong, Hollander, Kozaczynski, Brader, Delgado, Taylor, Wall, Slater, Imran, Cibraro, Cunningham)
How can a service prevent the disclosure of information about its internal implementation when an exception occurs?

Message Screening
(Hogg, Smith, Chong, Hollander, Kozaczynski, Brader, Delgado, Taylor, Wall, Slater, Imran, Cibraro, Cunningham)
How can a service be protected from malformed or malicious input?

Trusted Subsystem
(Hogg, Smith, Chong, Hollander, Kozaczynski, Brader, Delgado, Taylor, Wall, Slater, Imran, Cibraro, Cunningham)
How can a consumer be prevented from circumventing a service and directly accessing its resources?

Service Perimeter Guard
(Hogg, Smith, Chong, Hollander, Kozaczynski, Brader, Delgado, Taylor, Wall, Slater, Imran, Cibraro, Cunningham)
How can services that run in a private network be made available to external consumers without exposing internal resources?

Service Governance

Compatible Change (Orchard, Riley)
How can a service contract be modified without impacting consumers?

Version Identification (Orchard, Riely)
How can consumers be made aware of service contract version information?

Termination Notification (Orchard, Riley)
How can the scheduled expiry of a service contract be communicated to consumer programs?

Service Refactoring (Erl)
How can a service be evolved without impacting existing consumers?

Service Decomposition (Erl)
How can the granularity of a service be increased subsequent to its implementation?

Proxy Capability (Erl)
How can a service subject to decomposition continue to support consumers affected by the decomposition?

Decomposed Capability (Erl)
How can a service be designed to minimize the chances of capability logic deconstruction?

Distributed Capability (Erl)
How can a service preserve its functional context while also fulfilling special capability processing requirements?

Service Interaction Security

Data Confidentiality (Hogg, Smith, Chong, Hollander, Kozaczynski, Brader, Delgado, Taylor, Wall, Slater, Imran, Cibraro, Cunningham)
How can data within a message be protected so that it is not disclosed to unintended recipients while in transit?

Data Origin Authentication (Hogg, Smith, Chong, Hollander, Kozaczynski, Brader, Delgado, Taylor, Wall, Slater, Imran, Cibraro, Cunningham)
How can a service verify that a message originates from a known sender and that the message has not been tampered with in transit?

Direct Authentication (Hogg, Smith, Chong, Hollander, Kozaczynski, Brader, Delgado, Taylor, Wall, Slater, Imran, Cibraro, Cunningham)
How can a service verify the credentials provided by a consumer?

Brokered Authentication (Hogg, Smith, Chong, Hollander, Kozaczynski, Brader, Delgado, Taylor, Wall, Slater, Imran, Cibraro, Cunningham)
How can a service efficiently verify consumer credentials if the consumer and service do not trust each other or if the consumer requires access to multiple services?

Service Messaging

Service Messaging (Erl)
How can services interoperate without forming persistent, tightly coupled connections?

Messaging Metadata (Erl)
How can services be designed to process activity-specific data at runtime?

Service Agent (Erl)
How can event-driven logic be separated and governed independently?

Intermediate Routing (Little, Rischbeck, Simon)
How can dynamic runtime factors affect the path of a message?

State Messaging (Karmarkar)
How can a service remain stateless while participating in stateful interactions?

Service Callback (Karmarkar)
How can a service communicate asynchronously with its consumers?

Service Instance Routing (Karmarkar)
How can consumers contact and interact with service instances without the need for proprietary processing logic?

Asynchronous Queuing (Little, Rischbeck, Simon)
How can a service and its consumers accommodate isolated failures and avoid unnecessarily locking resources?

Reliable Messaging (Little, Rischbeck, Simon)
How can services communicate reliably when implemented in an unreliable environment?

Event-Driven Messaging (Little, Rischbeck, Simon)
How can service consumers be automatically notified of runtime service events?

Common Compound Design

Orchestration (Erl, Loesgen)
Co-existent application of Process Abstraction, State Repository, Process Centralization, and Compensating Service Transaction, can can be further extended with Atomic Service Transaction, Rules Centralization, and Data Model Transformation.

Enterprise Service Bus (Erl, Little, Rischbeck, Simon)
Co-existent application of Asynchronous Queuing, Intermediate Routing, and the Service Broker compound pattern and can be further extended via Reliable Messaging, Policy Centralization, Rules Centralization, and Event-Driven Messaging.

Service Broker (Little, Rischbeck, Simon)
Co-existent application of Data Model Transformation, Data Format Transformation, and Protocol Bridging..

Canonical Schema Bus (Utschig, Maier, Trops, Normann, Winterberg, Erl)
Co-existent application of Enterprise Service Bus, Decoupled Contract, Contract Centralization, and Canonical Schema.

Official Endpoint (Erl)
Joint application of Logic Centralization and Contract Centralization.

Federated Endpoint Layer (Erl)
Joint application of Official Endpoint, Service Normalization, Canonical Protocol, Canonical Schema, and Canonical Expression.

Three-Layer Inventory (Erl)
Joint application of Utility Abstraction, Entity Abstraction, and Process Abstraction.

Service Contract

Decoupled Contract (Erl)
How can a service express its capabilities independently of its implementation?

Contract Centralization (Erl)
How can direct consumer-to-implementation coupling be avoided?

Contract Denormalization (Erl)
How can a service contract facilitate consumer programs with differing data exchange requirements?

Concurrent Contracts (Erl)
How can a service facilitate multi-consumer coupling requirements and abstraction concerns at the same time?

Validation Abstraction (Erl)
How can service contracts be designed to more easily adapt to validation logic changes?

Capability Composition

Capability Composition (Erl)
How can a service capability solve a problem that requires logic outside of the service boundary?

Capability Recomposition (Erl)
How can the same capability be used to help solve multiple problems?