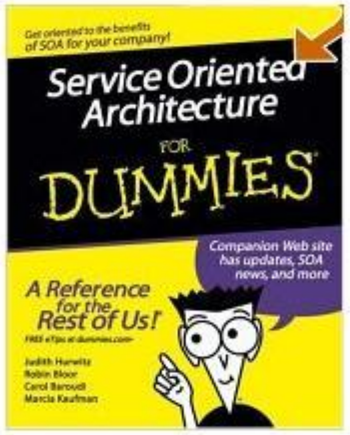
SOA

Service-Oriented Architecture

Book

- **SOA Made Simple**
  by Lonneke Dikmans; Ronald van Luttikhuizen (Dec 2012)
  Packt Publishing

  - Service Oriented Architecture for Dummies 2nd Ed., Jurwitz, Bloor, Kaufman, Halper. Wiley, 2009


Get oriented to the benefits of SOA for your company!
Service Oriented Architecture
FOR
DUMMIES
Companion Web site has updates, SOA news, and more
A Reference for the Rest of Us!
Judith Hurwitz
Robin Bloor
Carol Baroudi
Marcia Kaufman

Architecture solves IT problems relationally

Architecture

Service vs. product

- Product
  - what a manufacturer MAKES for a consumer
- Service
  - what a provider DOES for a consumer
  - can include making a product
    - milk
    - car with warranty
    - phone with service plan
    - internet service

Service in real life

- Breakfast : a service
  - The way consumers expect to get it: contract / SLA
    - hours: 5am to 8am
    - seating: open
    - average serving time: 10 minutes
  - Wait staff: interface
    - Menu: valid data types
    - ordering language, pre-formatted requests
    - china / take-out containers
  - Chefs: implementation

Architecture as a tool

- Architecture - helps IT / business alignment in
  - environment and component relationships
  - design and evolution principles
- The relationships of components
  - How rooms are sized and positioned for a house
  - How business units are assigned tasks
  - How a car's sub-systems are designed

Software architecture terms

- Component
  - any minimally exposed cohesive body of code
- Service
  - Something a customer sees as having value
  - Something IT sees having value

Software architecture terms

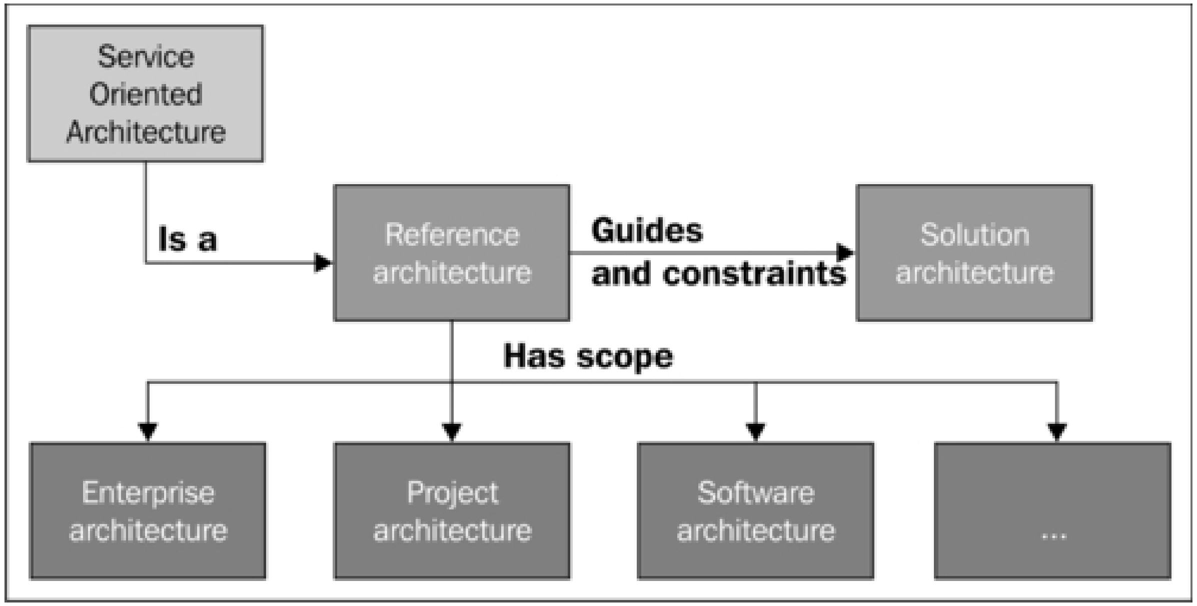
- Interface
  - what is exposed that is needed to communicate with that component
- Legacy systems
  - systems you wish were newer than they are

Software architecture terms

- Shared library
  - reusable sets of code in the same language
  - DLLs
- Modules
  - allows for lifecycle management
  - deployable during a running process
  - OSGI - https://en.wikipedia.org/wiki/OSGi , Java 9 in Mar 2017 (Jigsaw)

Reference architecture

- a template architecture for a similar domain
- standard vocabulary
- useful for building tools and sharing



Solution architecture

- a realized business need following a reference architecture
- Technical types
  - foundation - tool, product or service used
  - common systems - CRM, ERP, security
  - software architecture – microservices
- Scope types
  - industry - specific needs addressed
  - organization - finer grained than industry
  - project

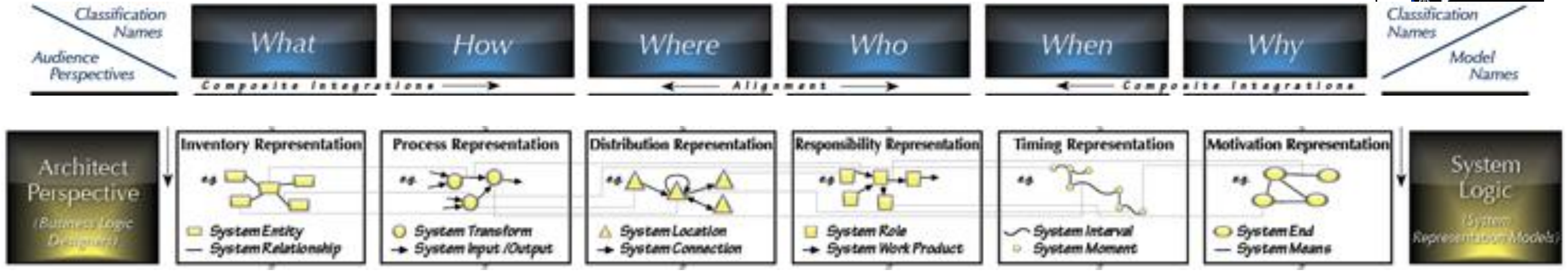
Architecture of SOA

- a reference architecture and a solution architecture
  - not well defined as a solution though
  - vendor controlled
- common uses
  - supply-chain networks
  - fast changing markets
  - many regulatory changes

SOA decoupling

- a reference architecture based on services
  - which are based on business processes
  - uses business descriptions
- data and logic should be separated
  - loosely coupled through services

Zachman framework



- Inventory
- Process
- Distribution
- Responsibility
- Timing
- Motivation

Management of IT services

Enterprise Analysis

Myths

- Every service has to be automated
  - can be human only also
- Every service is a web service
  - Can be a stored procedure or a class of methods
- Consumers of services are always IT systems

Business drivers

- Efficiency - Cost reduction
- Flexibility
- Standardization
  - XML (SOAP) – data format
  - REST or WS-* - application format
  - interoperability
- Shorter time to market (velocity)
- Increase in quality

Problem

- Two types of business IT problems
  - business and IT don’t speak the same language
  - workflow automation can cause
    - duplication of code for tasks
    - silos - isolation of tasks
    - scalability issues
- Service based orgs feel the pain more
  - main asset is information

Problem - language

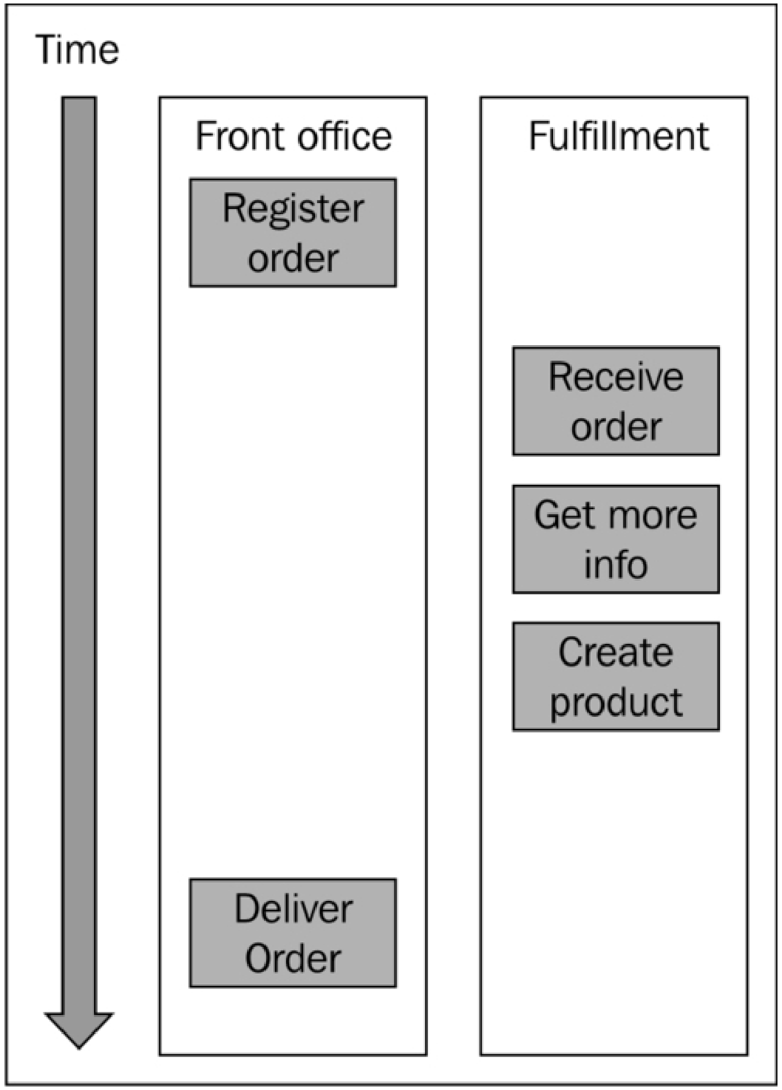
- business vs. technical language
  - best practice – use business language to describe problem
  - use cases, user stories
  - traceability to technical detail

Problem - duplication

- Workflow + data duplication
  - When organized by one (workflow usually), others can be easily duplicated
- Duplication possibilities
  - Data with validation rules
  - Data with stored procedures
  - Rules defining data
  - Workflow defining data
  - Workflow and rules not distinguished
  - Client / server / data base server

Problem - silos

- Isolated departments (functions)
- Organizational cohesion

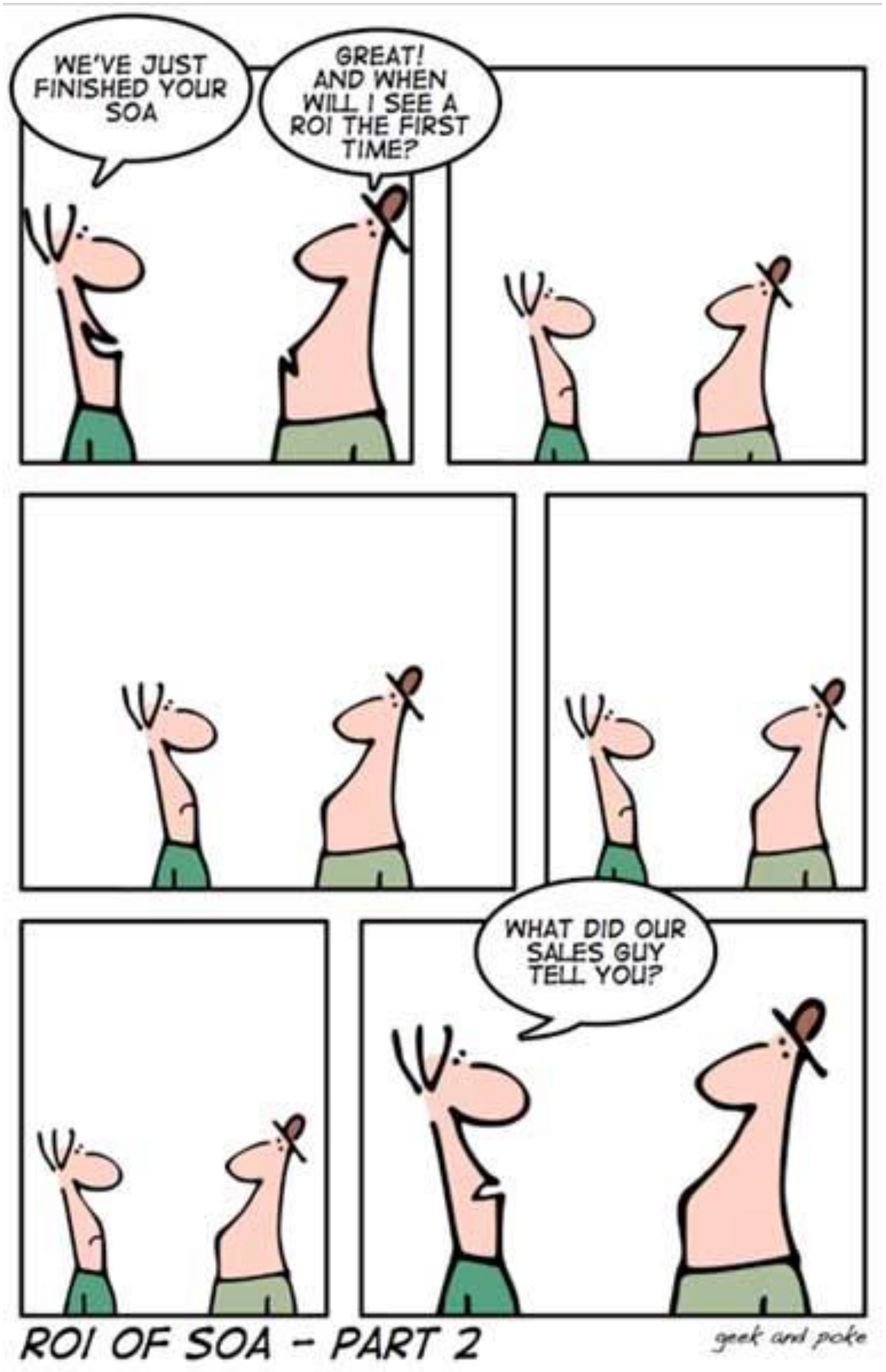


Problem - scalability

- Large distributed systems
  - are imperfect
    - Perfectionism spells P-A-R-A-L-Y-S-I-S - Winston Churchill
  - must deal with legacy systems
  - are heterogeneous (not homogeneous or standardized)
  - have more lifetime = more value
  - are complex
  - have different owners
  - have some redundancy - not all data in one place
  - with bottlenecks are dead

Problem - mismatched goals

- Business and IT alignment goal
  - Battlefield marketplace
  - Battlefield resource management
- Risks
  - Business - Is the workflow good / correct?
  - Organizational - Will people implement well?
  - Technical - Will it get to code correctly / fast enough?


WE'VE JUST FINISHED YOUR SOA
GREAT! AND WHEN WILL I SEE A ROI THE FIRST TIME?
WHAT DID OUR SALES GUY TELL YOU?
ROI OF SOA - PART 2
geek and poke

SOA
Service-Oriented Architecture

Analysis - segmentation

- Logical layers
  - topical breakdown to business, information, technical
  - security and administration cut across all
  - three perspectives: strategic, tactical, operational
- Logical tiers
  - software divisions of model (data), view (presentation), controller (logic)
- Views
  - stakeholder breakdown

Analysis – modeling languages

Language	Scope	Standard/ Proprietary	Target audience
Archimate	Business, information, and technology layer	Standard (The Open Group)	Architects, developers
UML(Unified Modeling Language)	Business, information and technology layer	Standard (OMG)	Developers, architects
BPMN (Business Process Modeling Notation)	Process models	Standard (OMG)	Process analysts, business consultants
EPC (Event driven Process Chain)	Process models	Proprietary	Process analysts, business consultants
ERM (Entity Relationship Model)	Information layer	De-facto standard	Developers, architects
Not applicable (Free format)	Business, information, and technology layer	Proprietary	Entire organization

SOA is one type of integration

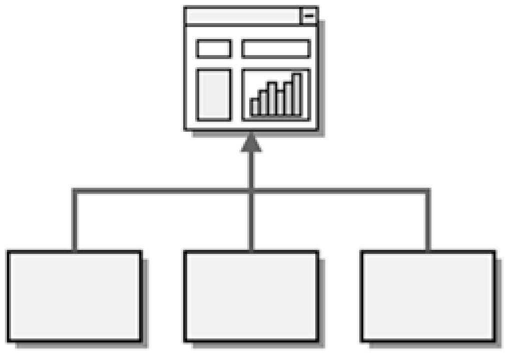
Integration design

Integration - a grey term

- Connecting computer systems, companies, or people.
- Six basic types
  - information portal
  - data replication
  - shared business functions
  - distributed business process
  - business to business integration
  - **service oriented architectures**

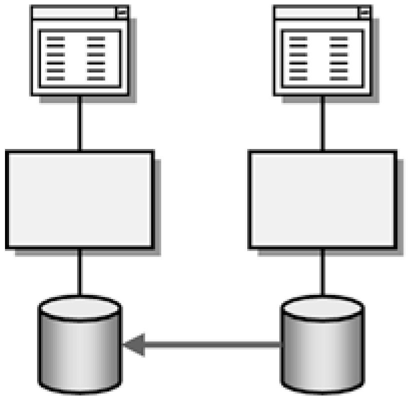
Information portal

- Aggregates data from multiple sources to a single display
  - Single sign on
- screens have zones for each system
- limited interaction



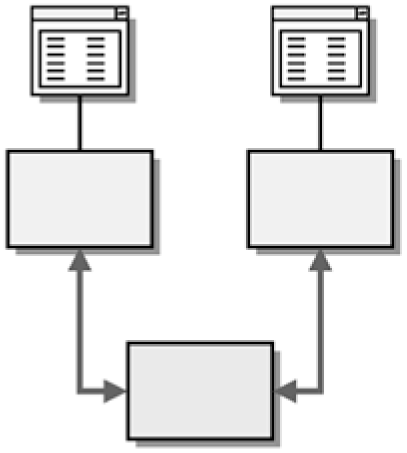
Data replication

- Multiple systems require access to the same data.
- Implementation types
  - replication functions in database
  - export and import data
  - transport data between systems with messages (MOM)



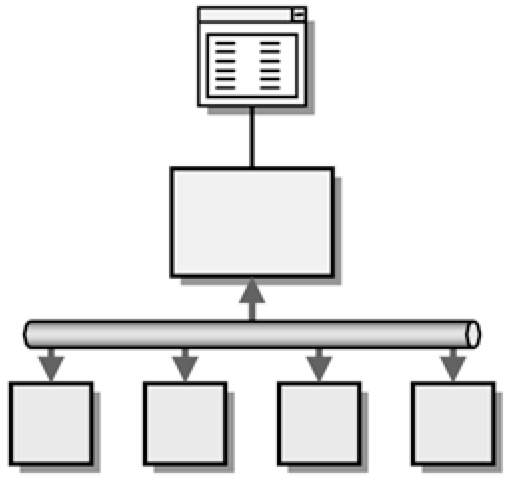
Shared business functions

- Besides sharing data, business wants to share functionality
  - business rules
  - Processing
- API/library management



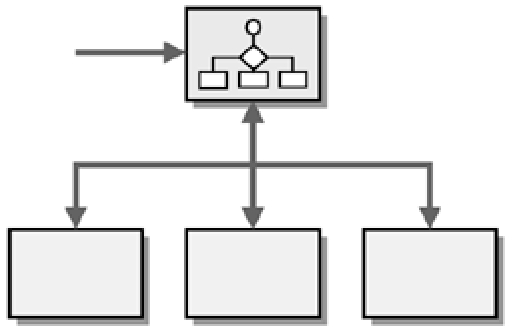
Service oriented architectures

- Shared business functions = (popular definition) services
- Blurs integration and distributed applications
- GUI --- business logic → services



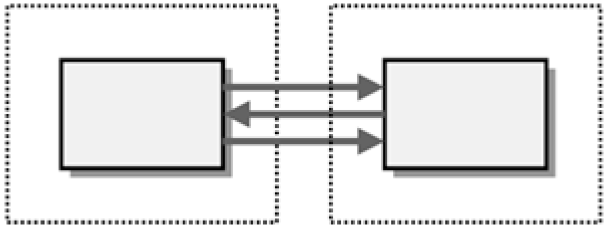
Distributed business process

- Business process management component manages business function execution across multiple systems
  - Orchestrated
  - Controller



Business to business integration

- A conversation between two enterprises
- Automated processes require standard formats
  - EDI
  - ebXML



Integration selection criteria

- Should you?
  - Where You Have to Distribute - Fowler
- Coupling - decrease dependencies
- Intrusiveness - minimize change and code
- Technology selection
- Data format - now and future
- Data timeliness - small chunks vs large

Integration selection criteria

- Data or functionality - sharing
- Remote communication - slower
- Reliability - hello?

Integration styles

- file transfer (coarse – COBOL, text, XML)
- shared database (schema enforced – SQL based)
- remote procedure invocation (function enforced - CORBA, RMI, COM, .NET remoting, AJAX).
- messaging (fine – Jini/Apache River, RabbitMQ)

Messaging

- Fowler - “we consider Messaging to be generally the best approach to enterprise application integration.”
- Asynchronous design is not taught
- Testing and debugging is harder

Integration styles - files?

- Just use files?
  - batched systems
    - COBOL
    - Unix - text
    - XML documents
  - very slow if too much or too many
  - no extra tools or integration packages

Integration styles - central data?

- Put data in one place and share?
  - keep everything in one database - coupled
  - can scale beyond capacity
  - better standards wise with SQL
  - schema design becomes more difficult with scale
    - integrations pose incompatibilities
    - read/write locking causes performance issues
  - distribution of data helps local access
    - design is problematic

Integration styles - method calls?

- Make it look like a local method call?
  - RPC / RMI
    - remote procedure call
    - remote method invocation
  - CORBA, DCOM, .NET Remoting, Java RMI
  - RPC style web services - WS-* (SOAP)
    - easy to pass through with web server
  - familiar to programmers but not a good fit

Integration styles - uncoupled?

- Make it completely uncoupled?
  - Send a **message** a let middleware handle it.
    - Frequent exchanges of small amounts of data is better.
  - Middleware can be ‘sneakernet’
  - More than two systems?
    - Use routing components - a message broker
  - Middleware needs a message endpoint
  - Most difficult style of integration
  - High cohesion (lots of work locally) + low adhesion (selective work remotely)

Implementation of the service itself

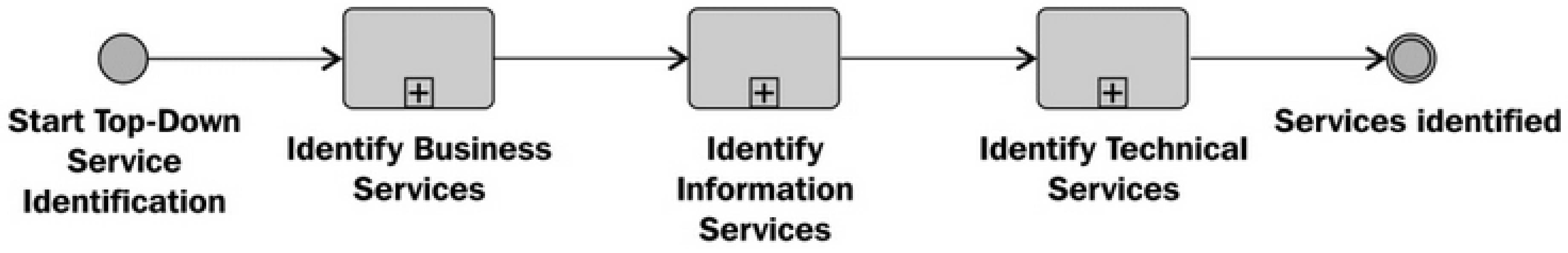
Design strategy

Service design strategies

- Three approaches
  - top down (contract first)
    - interfaces, code stubs, fill out stubs
  - bottom up
    - functionality, expose interfaces
  - meet in the middle

Identification processes

- Top down
  - stems from business services / EA
  - then current and planned services
  - then the service itself
- Bottom up
  - Inventory / baseline
  - Define the usable value needed and refactor as necessary


Start Top-Down Service Identification
Identify Business Services
Identify Information Services
Identify Technical Services
Services identified

Service identification

- Must have entry criteria of
  - business case
  - real project and real consumers
- Also called **service analysis**
  - Derived from requirements
  - Specifies functionality
- Iterative process

Service design principles

- Provide value
- Isolation – autonomous
  - Hides implementation
- Trust – security, fault prevention
- Idempotency
  - statefulness
  - repeatable with same outcomes
- Reusable, interoperable - plug-n-play

Isolation

- Isolation is a convenience for the consumer so that there are no dependencies/risk to the consumer
- Isolation does not depend on other services
  - JIT ordering vs mail ordering

Composition

- A bus transfer, plane changes/ flight legs
- Composition allows isolated services to be strung together
  - Assembly line at Chipotle
- Composition dependencies are internal
  - from this level to same level
  - from this level to lower level

Trust

- Factors that improve consumer motivation to use
- Security
  - message level security
  - stated explicitly in contract
  - handled by standards in implementation if possible
- Fault prevention and handling
  - business faults
  - user input faults
  - technical faults

Consumer reuse

- Consumers must trust the service quality
- Consumers must agree with the contract
- Consumers like transparent interfaces
  - I'll have the special and coffee.
  - I'll have coffee and the special.
  - I'll have the special. I'll also have coffee.

Idempotency & statefulness

- Which is better? (hint: introduce an asynch fault)
  - increaseSalary($100)
  - setSalary(originalEventTime, currentSalary + $100);
- Is this a problem?
  - Client visits web site. Signs-in.
  - Web server 1 in cluster receives request for sign-in. Approves. Sends cookie. Remembers user.
  - Client visits restricted page. Sends cookie.
  - Web server 2 in cluster receives request for page. Does not know user. Disapproves and requests sign-in

Reusability

- Software maxim: DRY (Don‘t repeat yourself)
- Requires management over parts that are reused
- Business rule consistency
- More reusability = more cohesiveness
  - tendency towards smaller services

Design paradigms

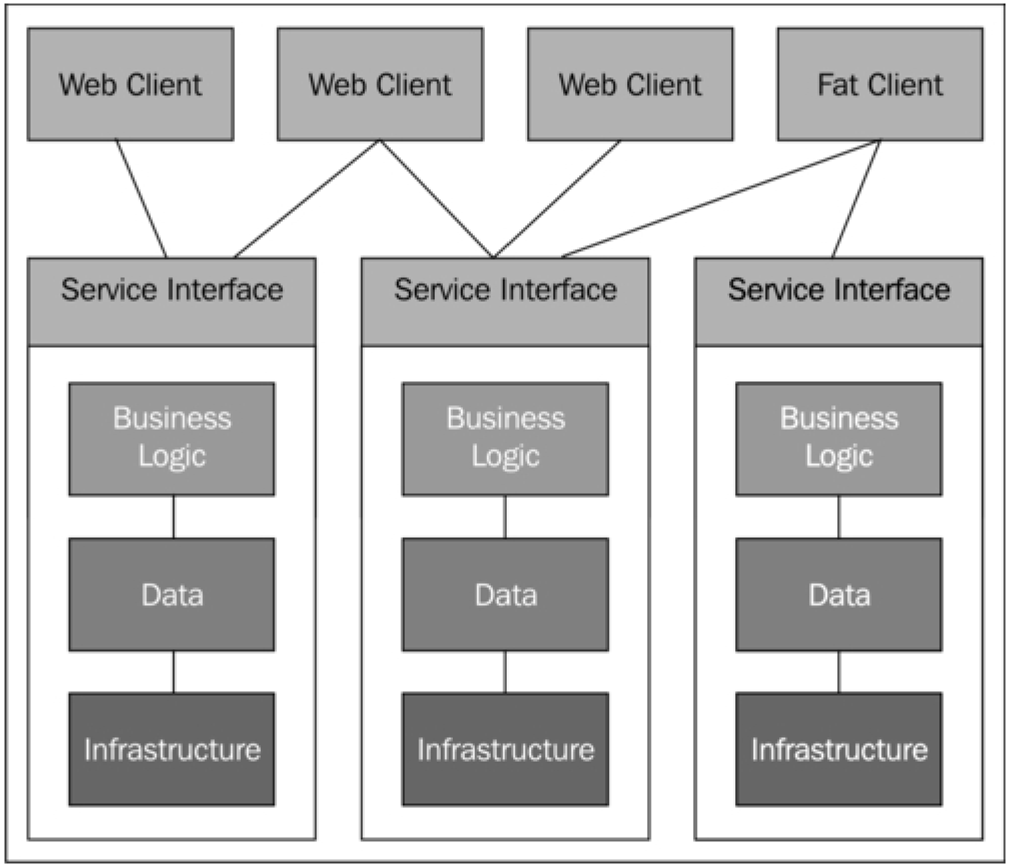
- Paradigm = Not an ideal, two related choices to work out a compromise
- Coupling vs cohesiveness
- Large vs small scope granularity
- Sync / Async
  - Reactive extensions - for data streams mostly
    - https://msdn.microsoft.com/en-us/data/gg577609.aspx
    - https://github.com/Reactive-Extensions/Rx.NET

Coupling / cohesion

- software term: loosely coupled
  - the need to change when another module changes
- In contrast to meaningful
  - software term: cohesive
  - a grouping of meaningful operations

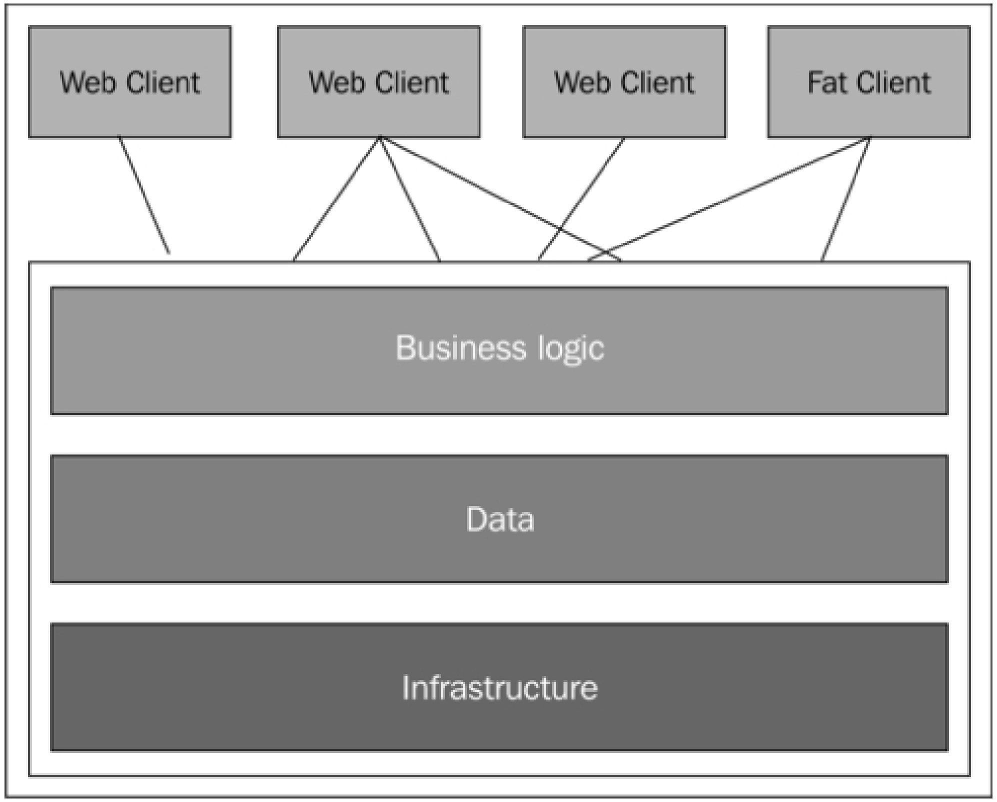
Coupling / cohesion

- High cohesiveness
- Low sharing of data & logic
- Partial solutions
  - Data centralization
  - CORBA



Coupling / cohesion

- Low cohesiveness as one component
  - must be pushed into logic as operations
- this is “implementing a service”
- it’s also the issue of branding
  - product line = operations



Granularity

- Grading based on
  - amount of scope - common
  - amount of value - associated with priority in PM

Granularity - three sizes

- More info in classification
- Elementary services
  - made up of simple operations
- Composite services
  - combinations of services
- Process/business service
  - full workflow of services

Service composition

- Create composite services with
  - an Enterprise Service Bus (ESB)
- Create process services with
  - Business Process Management (BPM) tooling (strict)
  - Case Management tooling (flexible)

Granularity - right sized

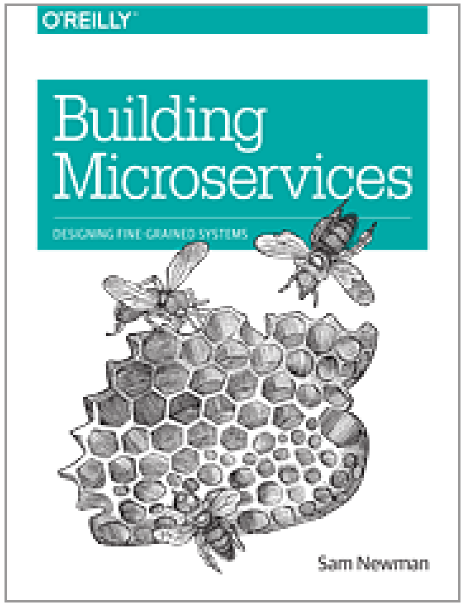
- Big enough for value
  - the largest service should be defined by how easily it can be managed by people
- Small enough to change implementation easily
  - the smallest operation should be able to be defined in business terms as the smallest task of a workflow
  - write out the workflow as a series of business tasks to understand
    - workflows are written without any mention of automation!

Synchronous/ Async

- The request and reply contain data
  - input and output
- Synchronous
  - Next service must wait for current service that triggered it to finish
- Asynchronous
  - Multiple services are triggered as fast as possible and complete when they want.
  - A callback is a synchronous functional response to an asynchronous trigger

Microservices

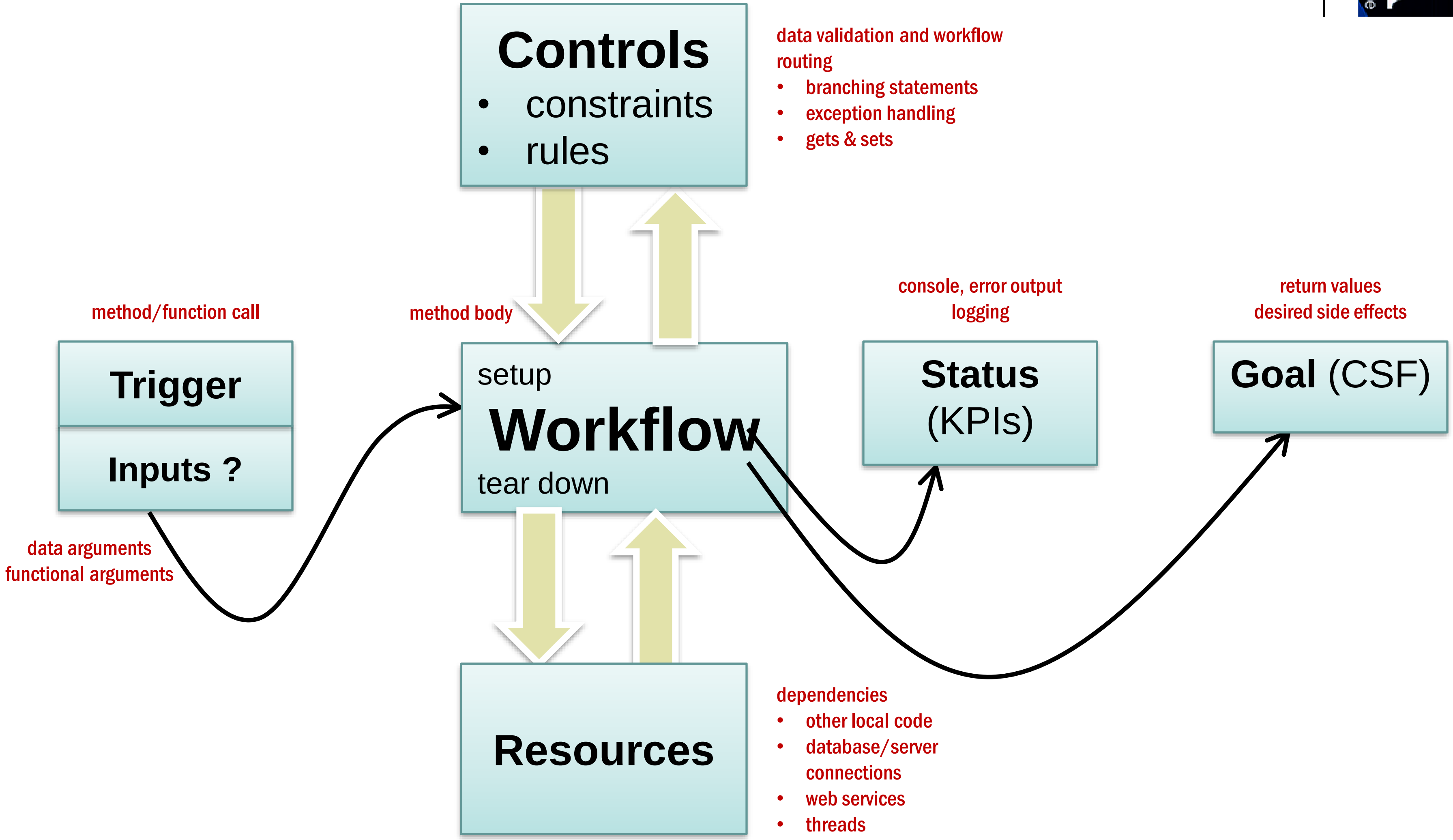
- small, autonomous services that work together - Sam Newman
- implements from the application ground up rather than designs from the enterprise top down
  - Agile : Scrum :: SOA : Microservices
- **Building Microservices**
  by Sam Newman, 2015



Service design

A process/service model

the process parts in computer language



Contracts and policies

- Contracts
  - How to trigger/call the service, how to parse the data that is returned, quality of service, costs
- Policies (sets of rules)
  - enforces security, logging…
  - added by message moving through a gateway - SSL
  - added by message coming from service under the control of an agent

Service contract

- Design the contract – business requirements
  - Define who is allowed to use the service and who can use what operation.
  - Decide how often the service is available.
  - Define the load the service should be able to handle.
  - Define other relevant **quality of service** attributes.

Service contract

- OrderService
  - OrderService will be capable of handling up to 2,000 orders an hour
  - OrderService will have an uptime of 99.95% and an outage will take no more than 90 minutes
  - Usage of OrderService is free of charge
- System non-functional requirements
  - capacity
  - availability

Business rules

- Derivations / process flow require the most modeling
  - encapsulations, relationships
  - data constraints - validations

	facets			result
applyCustomerDiscount	order total	status	before Dec 1?	percentage
	above or = $100	gold	yes	22%
			no	18%
		silver	yes	19%
			no	15%
	below $100	gold	yes	21%
			no	17%
		silver	yes	18%
			no	13%

Business rules

- One time use
  - can be specified in a functional requirement if small
  - Data validations or workflow routing
- Reusable
  - can be exposed as a service
  - specified in a rules document and referred to in requirements

Service classification

Classification types

- By actor (role of client)
  - who is responsible for triggering the service?
  - sometimes the system itself is the trigger by a timer - a cron job
- By security level
  - same as actor
- By channel
  - the communication service used to transmit the trigger

Classification types

- By organizational function
  - B2B or B2C
- Architectural layer
  - business, information, or technical
  - can be subdivided by granularity

Granularity

- Compounding services into bigger services
- By granularity - best!
  - elementary
    - commonly an entity class in OO programming
  - composite
    - commonly called controller classes in software
  - process
    - commonly a combination of business and system processes
- Seen as <<include>> or <<extends>> in use cases

Elementary services

- made up of simple operations
  - basic business task that should be completed together
  - dividing up the tasks will violate isolation principle
  - does not maintain state
- Software: class
  - operations are methods
- **Calculator** - add( ), subtract( ), multiply( ), divide( )
- **Web site** – each page, no cookie, no session state

Composability

- the ability to create new services by combining existing services
- Composition
  - the use of one service in another
  - t1 → t2 → t3 → Service → t4 → t5
- Aggregation
  - usually a straight-line process of multiple short-running services
  - t1 → Service1 → t2 → Service2 → t3

Composability

- Orchestration
  - complex path process of longer running services.
  - t1 → Service1 → t2 → Service2 → t3
    - if t3 is 1 → Service3
    - if t3 is 2 → Service4
    - if t3 is 3 → Service5
  - t4
  - do Service6 until complete
  - t5

Composite services

- Combinations of services
  - does not maintain state
  - typically straight-through (automated) process
- **Calculator** - combined **Arithmetic** functions for result, chained calculation
- Similar to a manager who uses staff for processes
  - better than consumers working directly with a large staff

Process/business service

- Full workflow of services
- Checkbook balancing - use **Calculator** to get expected result, compare to bank balance, if not balanced search for bad entry
  - can involve transaction processing
  - required to maintain state

Integrated UX

- Usually called a mash-up
- Portals / portlets
  - handles security and single sign-ons

SOA interface

The trigger

- A **consumer** triggers or uses a service
- A **client** makes a decision to trigger or use a service
- Business usage is the same
- System usage applies the consumer to system and the client to the business owner of the system

Event triggers

- Triggers
  - when data changes
  - when data is created
  - when data reaches a threshold
  - when a service is used
  - when a rule is invalid
- Publishers make events

Events (messages)

- Services can ‘subscribe’ to a type of event or ‘listen’ for that event
  - register a button with the click handler
- Events are states sent to other listening/subscribing services or internal mechanisms on a trigger
  - Button was clicked
  - Invoice was approved
- Events are detected by services or by event handlers (listener)
  - button.onClick(doSomething)

Event types

- Internal events
- External events
- Elementary events
  - single action - GUI
- Complex events
  - a business rule for event creation

Message routing - components

- Event providers (handlers) manage routing of events
  - Message brokers
  - Application servers
  - Enterprise Service Bus (ESB)
  - Message Oriented Middleware (MOM)

Message routing - queuing

- Event providers manage incoming messages by
  - queuing - message put in queue and delivers or consumer ‘pulls’ messages (1-1)

Message routing - pub/sub

- Event providers manage incoming messages by
  - pub/sub - message put in subscription queue and consumer ‘pulls’ available messages. Lossy. (1-*)

Service interface

- Design the interface – technical requirements
  - Design the operations and the functionality the operations offer.
  - Design the parameters of the operations.
  - Design the return value or the effect of the operation.
  - Design test cases for the operations.

Service interface

- Points of contact between consumer and service
  - includes a payload, operation name
- Functional interfaces
  - summary of publicly available capabilities
  - name, input, output
  - faults (error conditions)

Service Interface (API)

Operation	Description	Input	Output	Business fault
Order product	Creates a new order in the Order System and returns the calculated price and new identification for the order. All inputs are required. If the customer ID is invalid, a customer not found fault is returned. If the product is not in stock, a product not in stock fault will be returned.	Product identification, Quantity, Customer identification	Order identification, total price	Customer not found, product not in stock
Retrieve order information	Retrieves the order information from the Order System. All inputs are required. If the order is not found based on the order ID, an order not found fault would be returned.	Order identification	Order date, product identification, quantity, customer id, total price	Order not found

The messaging interface

- How do you send packets of data?
  - message via message channel
- How do you know where to send the data?
  - Use a message router if not known
- What data format do you use?
  - A message translator will convert and forward
- How do I connect my application into the system?
  - Implement endpoints for sending and receiving

Service interfaces

- Interface types
  - proprietary - SAP, Oracle E-Business, ERPs…
  - web - WS-* (SOAP), REST
    - HTTP – GET request…, AJAX (XMLHttpRequest - XHR)
    - WebSockets - direct connection, fastest to use
    - PeerJS - http://peerjs.com/ (WebRTC DataChannel, RTCPeerConnection) – peer-to-peer
    - Server-Sent Events EventSource API (server push)

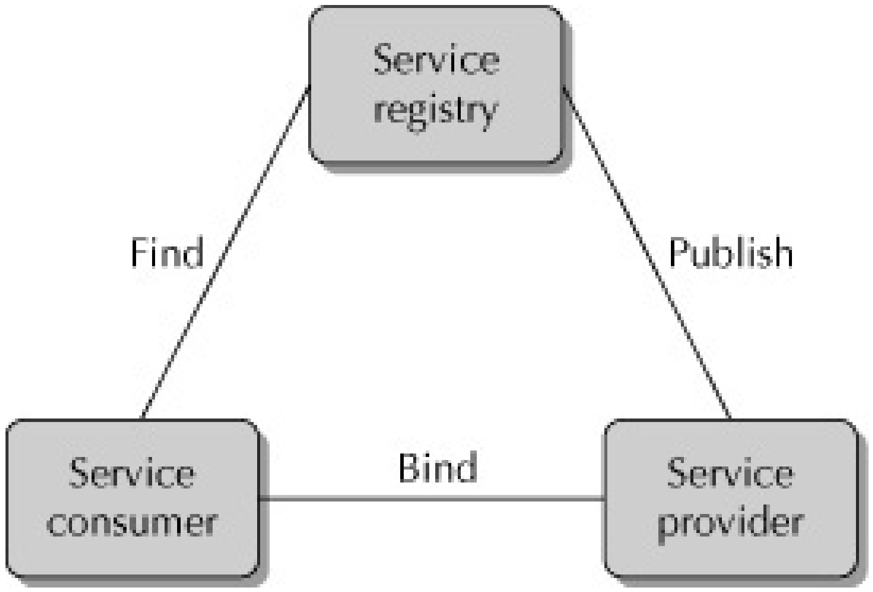
Service infrastructure

Canonical Data Model

- A data dictionary/catalog for services
- The schema to publish service data
  - WSDLs – WS-*
  - proprietary data schemas (XSDs)
  - vertical market data schemas (HL7, XBRL, RSS)
  - Public schemas – http://schema.org
- Transformations
  - XSLT

Service registry

- WS-*
- Breakfast services
  - Yelp
  - Telephone directory

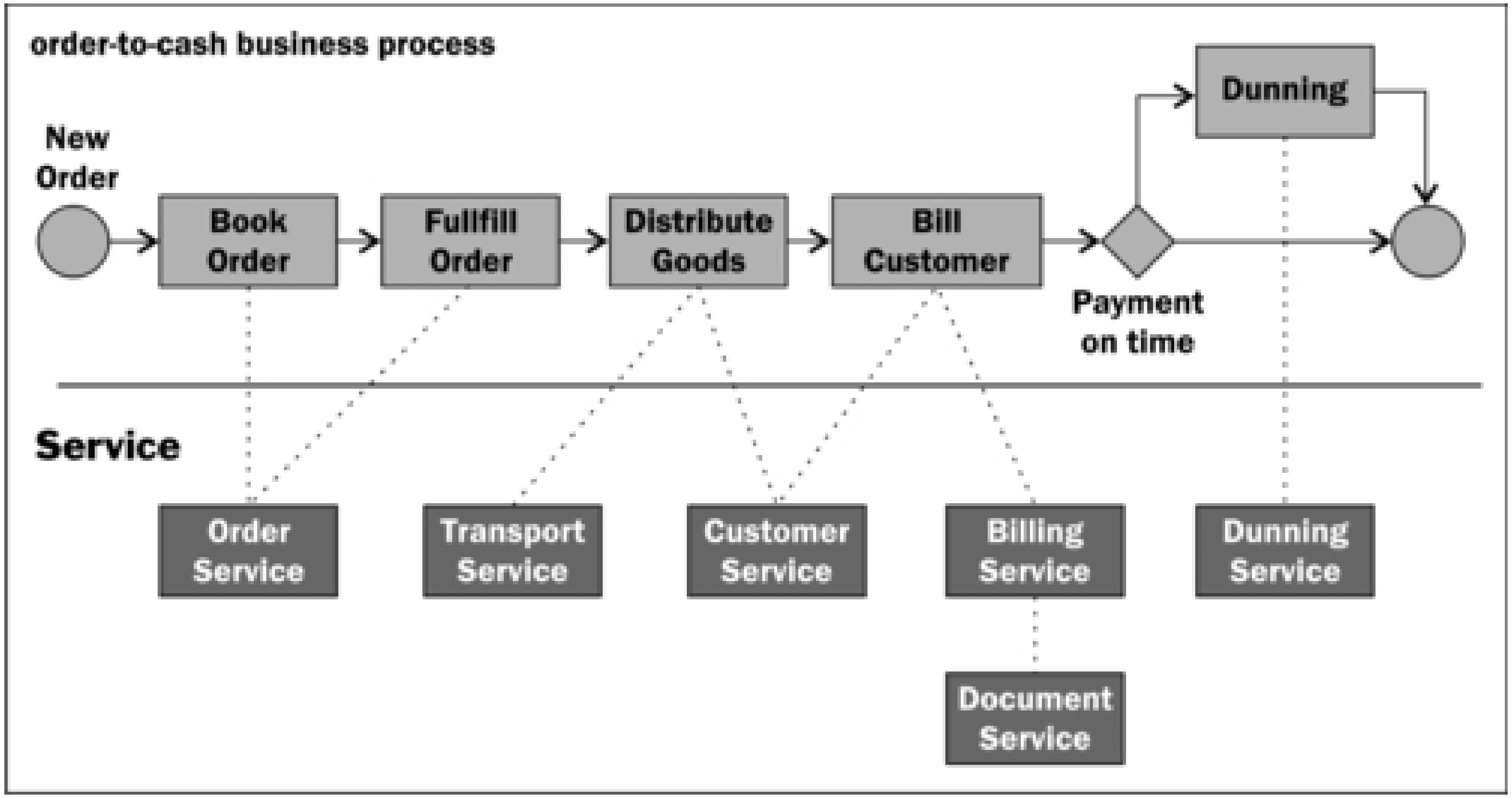


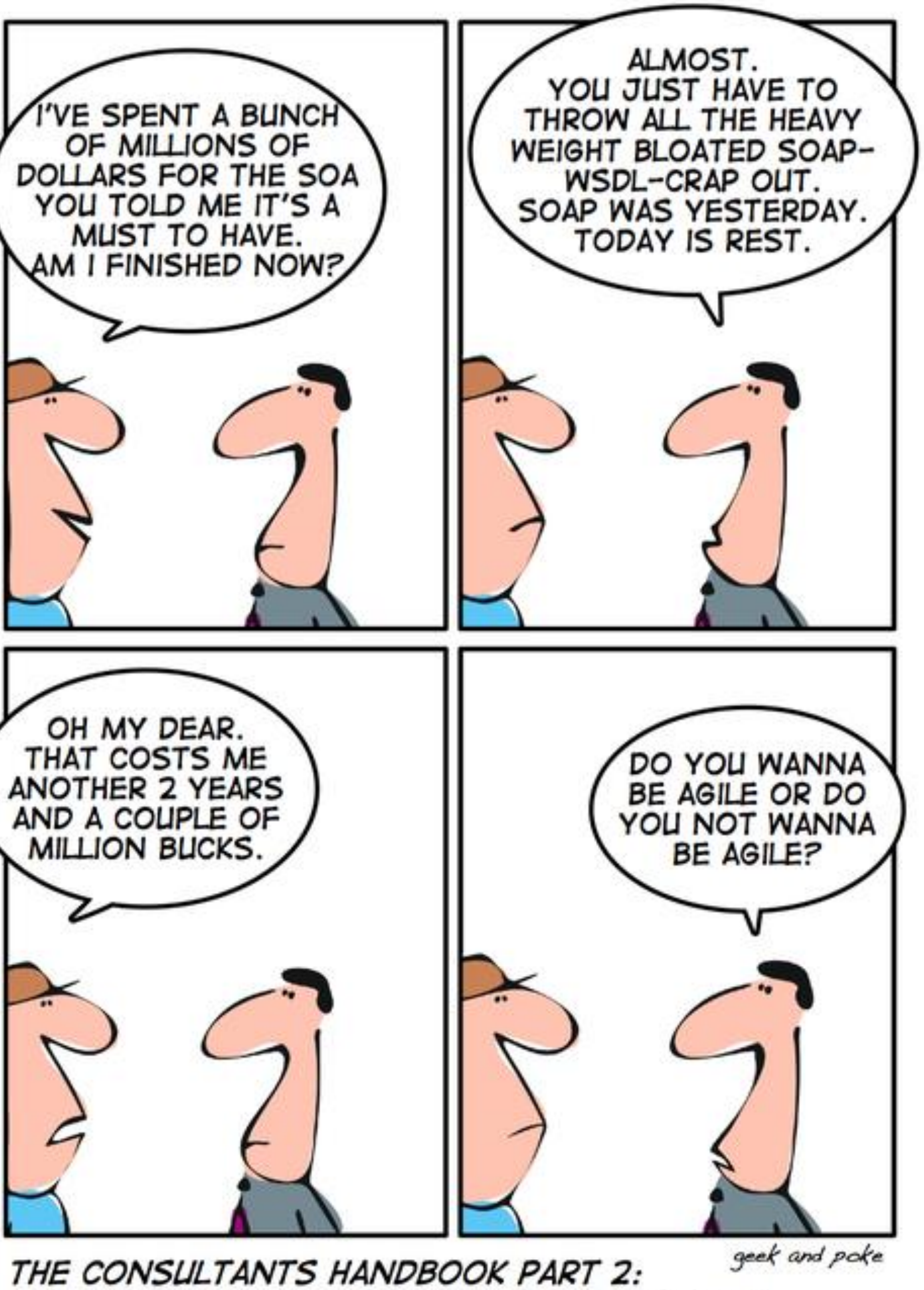
Case management

- Workflow driven by knowledge outcomes
  - processes are modeled with rules
  - BPMN processes + ad-hoc tasks + content management + some kind of social capability
- Dynamic Case Management
  - A semi-structured but also collaborative, dynamic, human and information-intensive process that is driven by outside events and requires incremental and progressive responses from the business domain handling the case - ***Forrester***

Orchestration

- An **orchestration** (specific order) of services



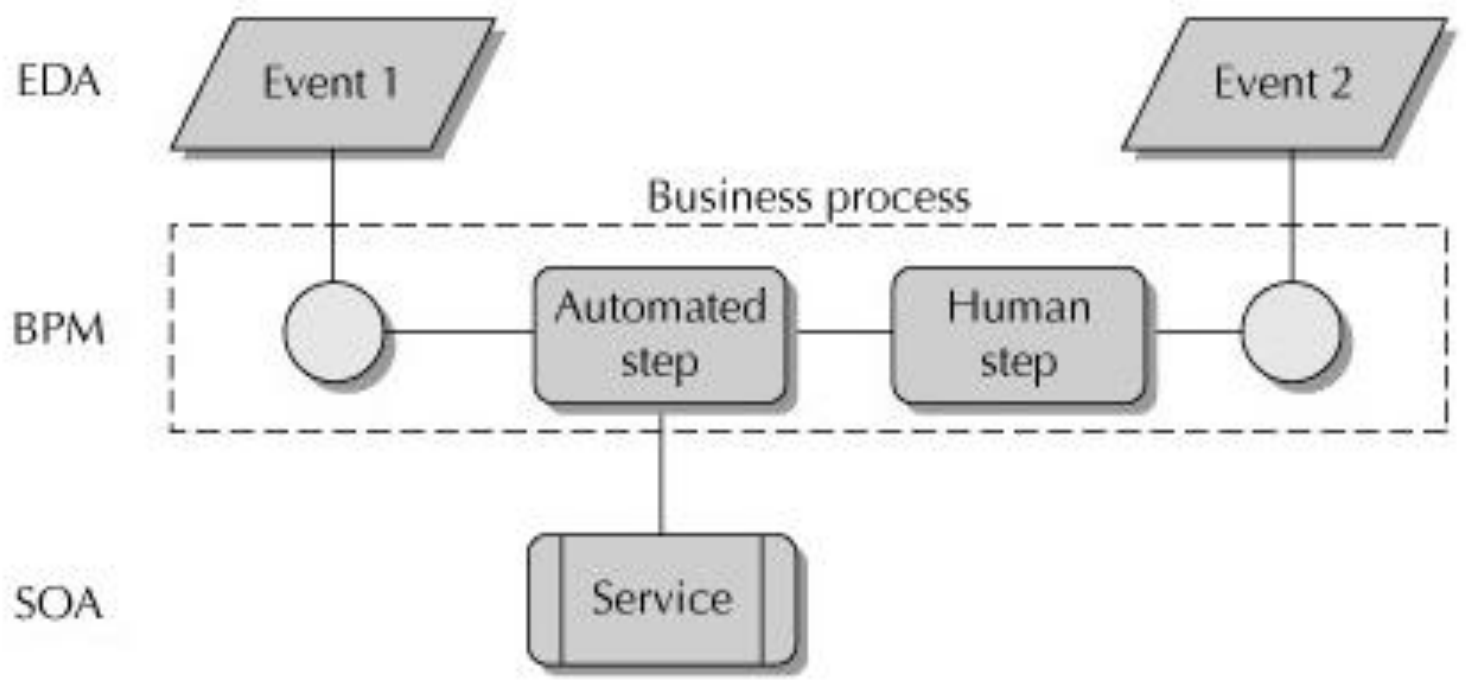

I'VE SPENT A BUNCH OF MILLIONS OF DOLLARS FOR THE SOA YOU TOLD ME IT'S A MUST TO HAVE. AM I FINISHED NOW?
ALMOST. YOU JUST HAVE TO THROW ALL THE HEAVY WEIGHT BLOATED SOAP-WSDL-CRAP OUT. SOAP WAS YESTERDAY. TODAY IS REST.
OH MY DEAR. THAT COSTS ME ANOTHER 2 YEARS AND A COUPLE OF MILLION BUCKS.
DO YOU WANNA BE AGILE OR DO YOU NOT WANNA BE AGILE?
geek and poke

THE CONSULTANTS HANDBOOK PART 2:
MAKE SURE YOUR COSTUMER IS ALWAYS AGILE

SOA
Service-Oriented Architecture

SOA, BPM, and EDA

- Event-Driven Architecture
- Business Process Management

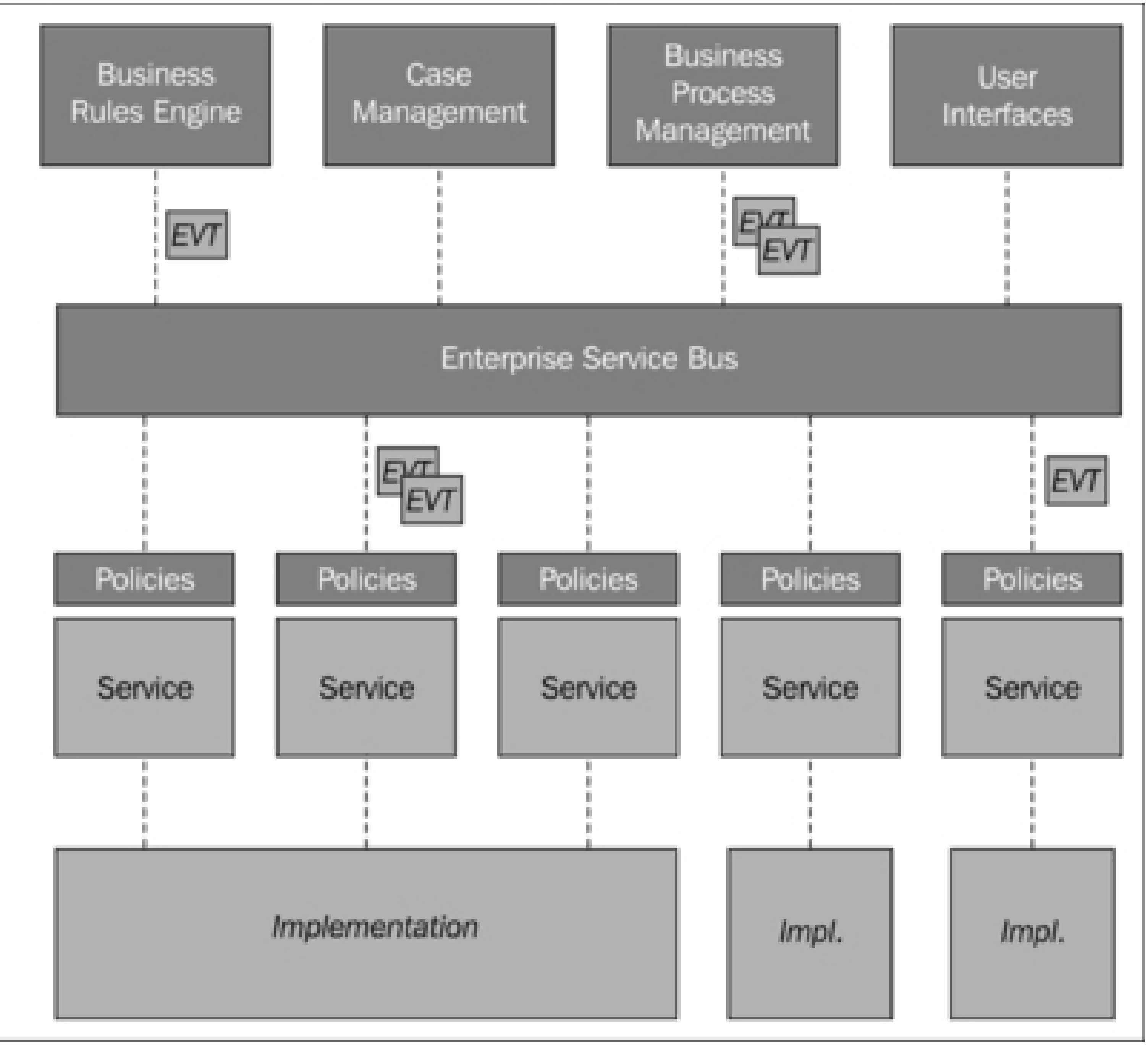


Service implementations

- Implementation
  - What is hidden from the consumer
  - Three ways to create
    - Use existing software, expose
    - Build
    - Combine existing into service

Enterprise Service Bus

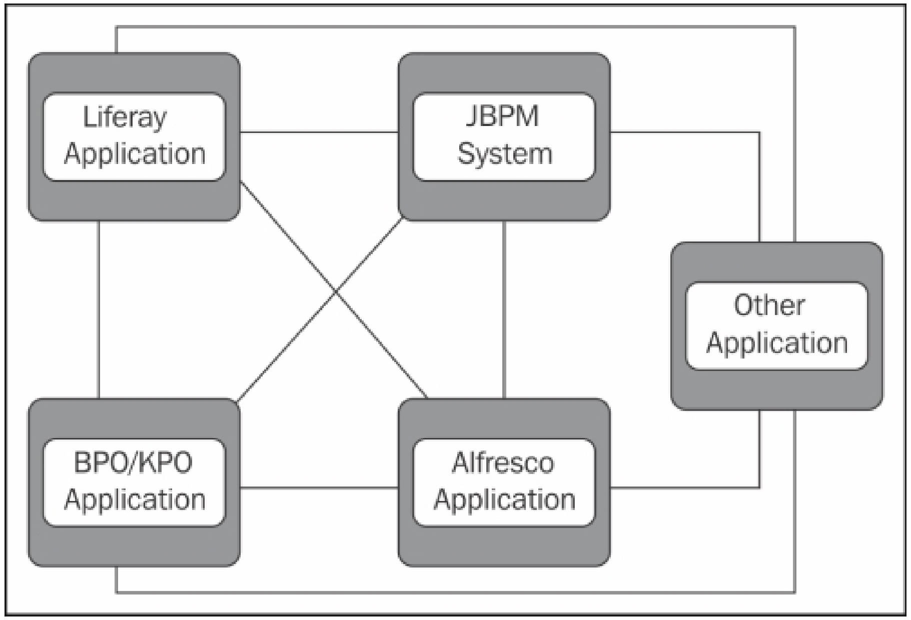
- Enterprise Service Bus
  - a connecting platform
  - originally the EAI bus
- ESB common functions
  - validation - input and output
  - composite service creation
  - routing - not based on changeable addresses, acts as a directory service
  - monitoring
  - transformation - protocols and data models

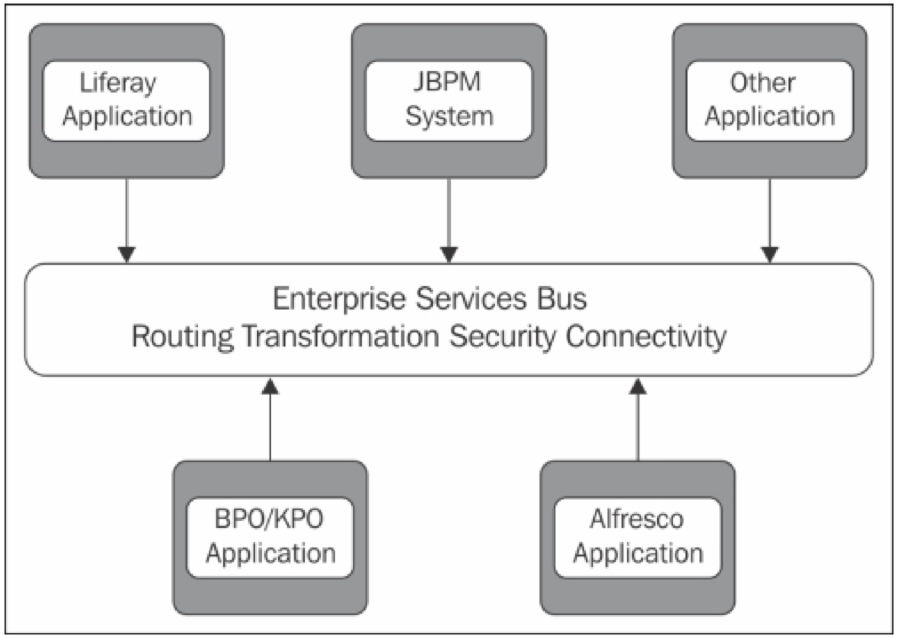

Business Rules Engine
Case Management
Business Process Management
User Interfaces
EVT
EVT
EVT
Enterprise Service Bus
EVT
EVT
EVT
Policies
Policies
Policies
Policies
Policies
Service
Service
Service
Service
Service
Implementation
Impl.
Impl.

Before & after ESB

SOA Service-Oriented Architecture

- The sales pitch





The magic bus

- Wow, only two steps away from where you want
- Scale up and you lose control
  - dependencies become lost
- Very attractive concept
  - Sonic and Tibco fight over who invented it
- Needs service structure
  - not just a path for interoperability

ESB advantages / disadvantages

- Advantages
  - decoupling - hide changes
- Disadvantages
  - complexity
  - requires good design

Service tools

Implementation

- Design the implementation
  - Design the components needed for the implementation.
  - Decide what tool or language to use for the implementation.
  - Decide what tools to test the implementation.

Implementation

- Must use a real language
- Your limited view of the implementation
  - Menu item #7
- Your actual implementation
  - Wreck a couple of cackleberries on a raft with wax
- Bound by constraints of knowledge and skills of developers, time, money, previous implementations

User interface

- Can be a service
- Is a consumer of services
  - creates a more integrated UX
- Capabilities
  - location and device transparency
  - mash-ups (multiple services)
  - contextual (relevant services shown together)
  - event-driven, customizable, task-enabled, collaboration
- HTML5 / CSS / JavaScript or jQuery

Security

- Identity and Access Management (IAM) tooling
  - Identity management - looking up members in Windows Active Directory or other LDAP service
  - Authenti**cat**ion - who gets let in
    - Single sign-on - one challenge for multiple systems
  - Author**iz**ation - who gets access to resources in there
- Messages require authentication also besides people

WS-* WSDL

- Web Service Description Language
  - a generic language for the service contract
- WSDL document
  - describes a service
  - <wsdl:**operation** name="orderProduct">
<wsdl:**input** message="ord:OrderProductRequestMessage"/>
<wsdl:**output** message="ord:OrderProductResponseMessage"/>
<wsdl:**fault** message="ord:ProductNotInStockFaultMessage" name="ProductNotInStockFault"/>
<wsdl:**fault** message="ord:CustomerNotFoundFaultMessage" name="CustomerNotFound"/>
</wsdl:operation>

Sample web service

- WSDL
  - http://ws.cdyne.com/ip2geo/ip2geo.asmx?wsdl
- POST implementation using <form>
  - http://ws.cdyne.com/ip2geo/ip2geo.asmx?op=ResolveIP

Web service - GET

- Google.com = 216.58.210.14
- GET address
  - http://ws.cdyne.com/ip2geo/ip2geo.asmx/ResolveIP?ipAddress=**<ip>**&licenseKey=0
  - http://whois.net/domain-name-ip-address/ can be used to translate domains to IPs

WS-* Services

- Sending data
  - SOAP - first popular protocol
- WSDL - service description
  - UDDI - registry protocol, used internally if at all
- WS-Security
- WS-RM

RESTful services

- **RE**presentational **S**tate **T**ransfer
- based on original concept of the web
- Sending/receiving data
  - SOAP, XHTML, XML, text
  - JSON - JavaScript protocol, favored because it's simpler
- request types
  - GET, PUT, POST, DELETE, PATCH
  - same as a database set of basic operations

REST case studies

- Netflix moved monolithic tiers to cloud services
  - Open-source service registry (Eureka), gateway (Zuul) and others. See their engineering blog and the Netflix OSS GitHub repository.
- LinkedIn moved monolithic app to distributed services.
  - This InfoQ presentation from Jay Krepps is a great overview.
- Twitter switched from "monorails" app to distributed JVM based services.
  - Jeremy Cloud discusses this in an InfoQ presentation, "Decomposing Twitter".

REST JSON examples

- https://www.mediawiki.org/wiki/Special:ApiSandbox
  - #action=opensearch
  - &format=json
  - &search=Te

JSONP vs CORS

- Techniques to request cross-domain but still prevent cross-domain scripting attacks
- http://en.wikipedia.org/wiki/Same-origin_policy
- JSON + padding
  - GET only
- Cross-origin resource sharing
  - all REST

.NET

- (4.5) WEB API
  - supercedes WCF REST and ASP.NET AJAX with ASMX
    - WCF better for WS-* SOAP based
  - based on MVC extensible Formatters and Filters
    - better for request / response
  - Better testing ability
- Nancy

Java

- Java API for XML (JAX)
- Jersey - RESTful service and client development
- Axis2 – SOAP + REST
- Restlet SAS

Design tooling

- BPM tools
- UML models
  - Visio
- Archimate - http://www.opengroup.org/subjectareas/enterprise/archimate
  - Archi - http://archi.cetis.ac.uk/

Development tooling

- IDEs
  - Visual Studio, Eclipse
- XML
  - XMLSpy
- Languages / Frameworks
- Testing
- Source control
  - Git (use Atlassian’s SourceTree), TFS, Subversion, CVS

Database tooling

- Automatic REST API platforms
  - Google Firebase - https://firebase.google.com/
  - DreamFactory - https://www.dreamfactory.com/
  - Loopback (JS) - https://loopback.io/
  - Database to API (PHP) - https://github.com/project-open-data/db-to-api

Service registry / repository

- service registry
  - used to describe services
  - searchable
  - the yellow pages of the services
- service repository
  - used to store artifacts of services
- need?
  - how many services do you have?
  - no useful public registry

Implementing

Solution architectures

BPM

- Business Process Management
- the lifecycle
  - model → simulate → implement → execute → monitor → optimize
  - software (analysis, design, code, deploy, maintain, refactor/improve)
- languages
  - Business Process Modeling Notation (BPMN)
  - Business Process Execution Language (BPEL)

Suites

- Oracle, IBM, and Microsoft
- Advantages
  - Tested and supported
  - Optimized
  - Documented
  - One vendor
  - Better support
  - Probably deal with that environment already

Suites

- Disadvantages
  - overbuying
  - compromising for features
- Build or buy
  - the best of breed allure
  - generally a Java EE or .NET solution
- Know your strategy first
  - know what you will need and why

Comparison

- Continually changing
  - read book for overview
  - then check out current offerings

Oracle

- Uses Java (owns it)
- http://www.oracle.com/us/technologies/bpm/suite/overview/index.html
- http://www.oracle.com/us/products/middleware/soa/overview/index.html

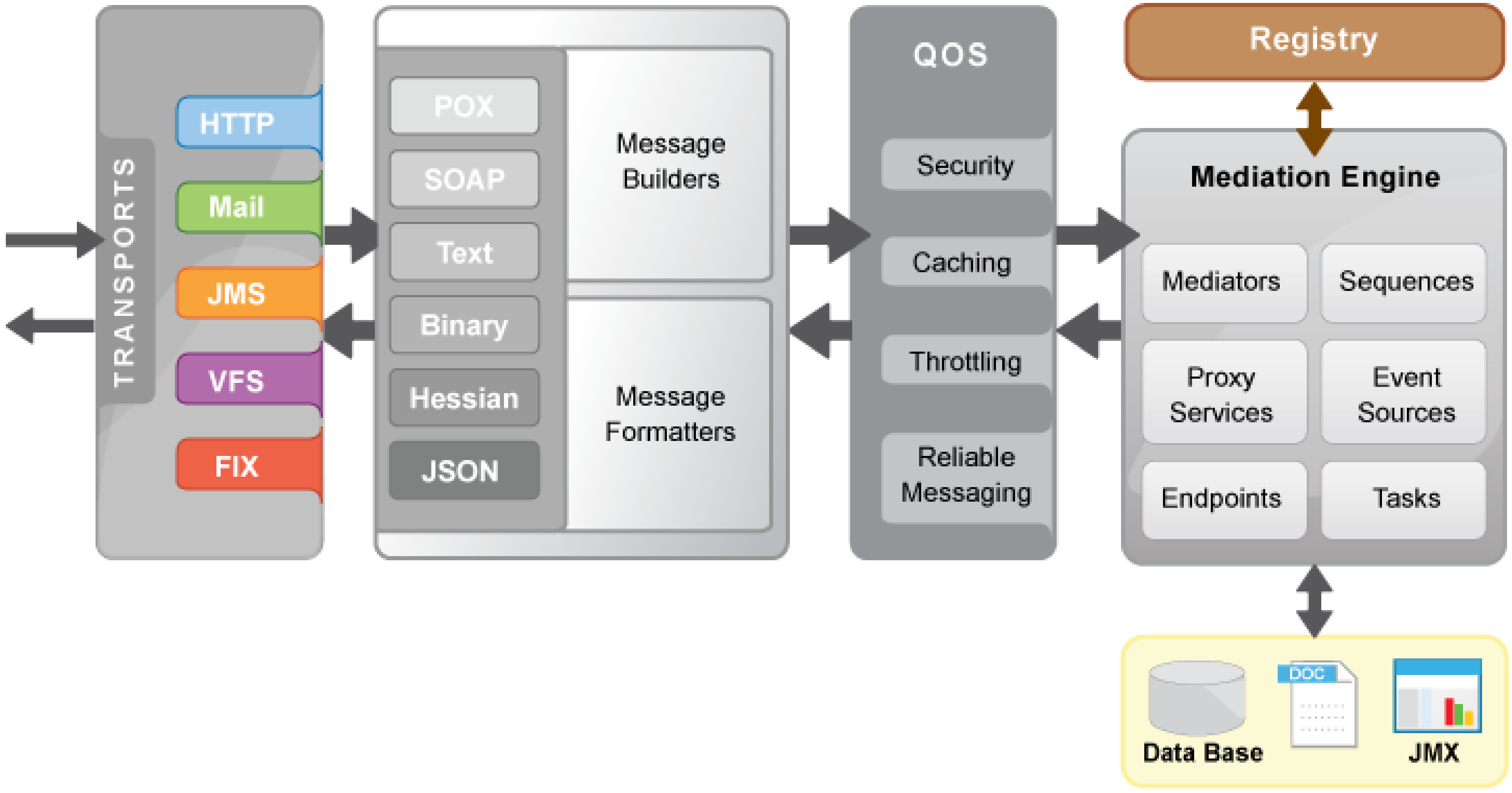
IBM

- Uses Java (wished they owned it)
- Websphere branded
- http://www-03.ibm.com/software/products/us/en/category/BPM-SOFTWARE
- http://www-01.ibm.com/software/solutions/soa/

Microsoft

- Trends / SOA
  - http://msdn.microsoft.com/en-us/library/bb977471.aspx
- WCF Data Services 4.5
  - http://msdn.microsoft.com/en-us/library/cc668792.aspx
- ESB toolkit for BizTalk Server

Apache Synapse ESB



WSO2

- http://wso2.com/landing/enabling-the-connected-business/
  - 20 products covering all major categories from integration and API management to identity and analytics
- Paul Fremantle, Recognized by InfoWorld as a Top 25 CTO
  - also developed Apache Synapse ESB

OData.org or schema.org?

- OData - http://www.odata.org/
  - mark up data
  - access data by services making queries
  - data is not in a relationship format
- Schema.org - http://schema.org/
  - mark up data in HTML
  - access data by requesting web page
  - transparent to browsers
  - reachable by search engines / services

Service metrics

Metrics

- Usage by service
- Usage by function
- Security access denials

Bus features

Testing

Changes to traditional testing

- Moving from small number of many functioned apps to large number of isolated functions
- More testing is done earlier.
- Requires more planning due to asynchronous nature, less grunt work
- Higher need for conformance to rules because of cumulative nature

Issues – traditional vs services

● Test tools invoke, expect return	● Invoking may not return when expected or at all
● System performance done at end	● Performance done as early as possible
● Tester skills are about functionality	● Tester skills need various technical knowledge
● Ability to control state, seeding of values	● Minimal state to manage
● One app, one priority, one set of risks	● Multiple clients, higher priority, more risk.

Test strategy

- Tests are
  - service
  - end to end / flow-through
  - operation interface
    - each could have different security
  - Non-functional
    - Security, capacity, performance,
    - time-related tests – asynchronous, idempotent
  - operation implementation (not needed if reuse)

Test design

- Interface
  - Operations and functionality
  - Parameters of operations
  - Return value or side effects
- Test activities
  - Test case documentation for operations - validation

Test design

- Contract
  - Who can use the service?
  - Who can use what operation?
  - How often the service is available?
  - What load the service can handle?
  - Any other quality of service attributes?
- Test activities
  - Security test
  - State test
  - Boundary test
  - Load test

Test design

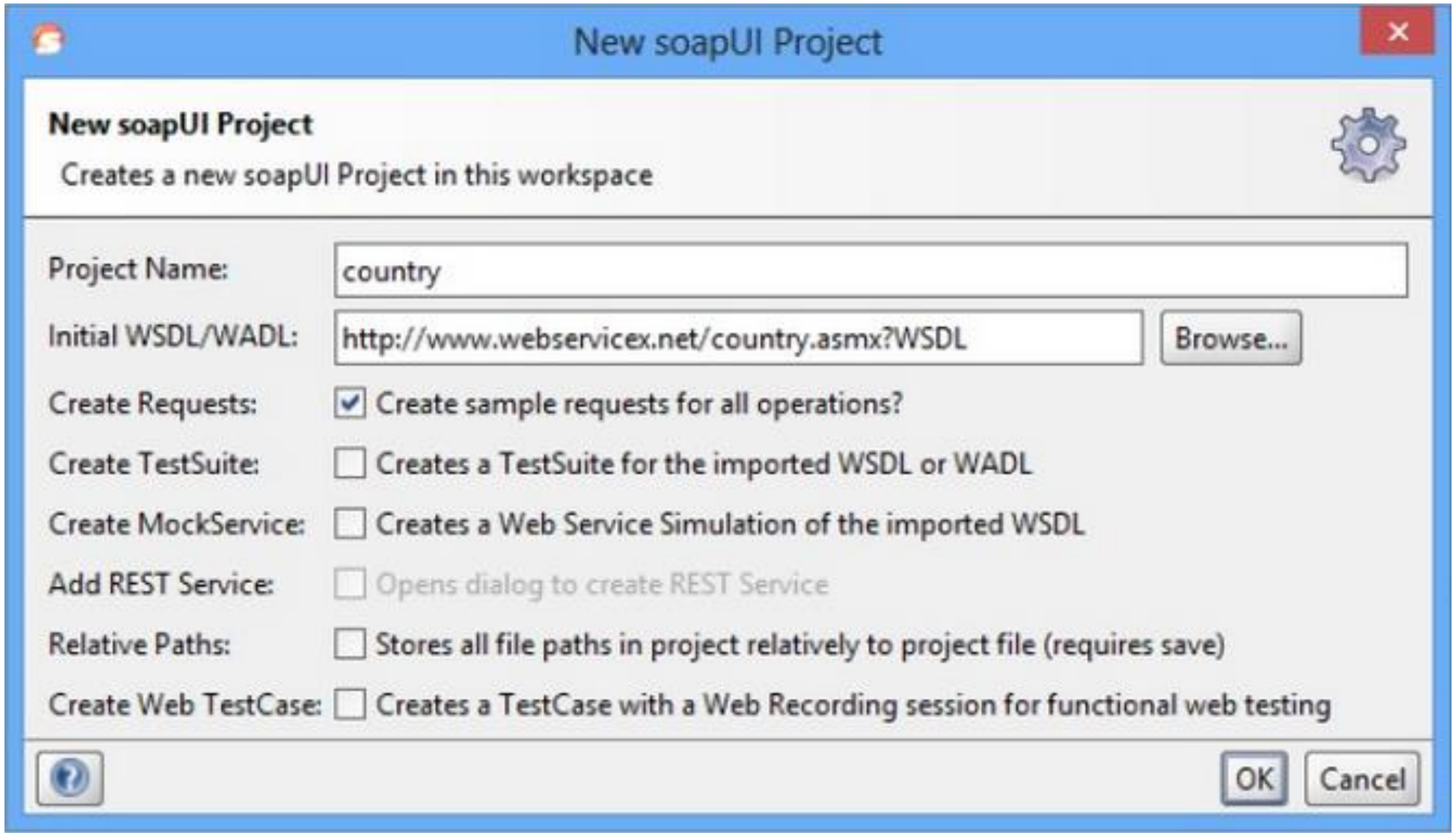
- Reusability
  - Check logs for use
  - Refactor

Test design

- Developers usually build mock Web service environments
- Emulate the myriad of client requests of server via test scripts including vulnerability tests
- Focus of new vendor test tools is to try to quickly and easily emulate any endpoint (client or server) of a Web service
- Point such tools to a valid WSDL and emulate both the client and server endpoints simultaneously
- Verify that the Intermediary (SOAP, security, etc.) handles the requests and responses as expected and policies are accurately reflected

Test tools

- SmartBear soapUI
  - http://www.soapui.org/



New soapUI Project

Creates a new soapUI Project in this workspace

Project Name:	country
Initial WSDL/WADL:	http://www.webservicex.net/country.asmx?WSDL
Create Requests:	☑ Create sample requests for all operations?
Create TestSuite:	☐ Creates a TestSuite for the imported WSDL or WADL
Create MockService:	☐ Creates a Web Service Simulation of the imported WSDL
Add REST Service:	☐ Opens dialog to create REST Service
Relative Paths:	☐ Stores all file paths in project relatively to project file (requires save)
Create Web TestCase:	☐ Creates a TestCase with a Web Recording session for functional web testing

Test tools

- Optimyz - WebServiceTester is an end-to-end product offering automatic test generation; functional, regression, and load testing; conformance testing against WS-I Profiles, BPEL-based orchestration testing; secure Web services testing; and debugging and diagnostics.
- Mercury(now HP) - "end to end" solution for Web services testing in the form of three offerings: LoadRunner, QuickTest Professional and Business Process Testing, its newest tool that sits on top of LoadRunner.

Test tools

- Empirix Inc. -- e-TEST: e-Manager Enterprise, test management; eTester, functional testing; e-Load, scalability testing.
- Parasoft -- SOAPtest, WSDL validation, unit and functional testing of the client and server, performance testing
- IBM Rational Software Co. -- TestStudio, unit, functionality, performance, and load testing; PurifyPlus, runtime analysis tool for detects memory and performance bottlenecks early in the development cycle

Test tools

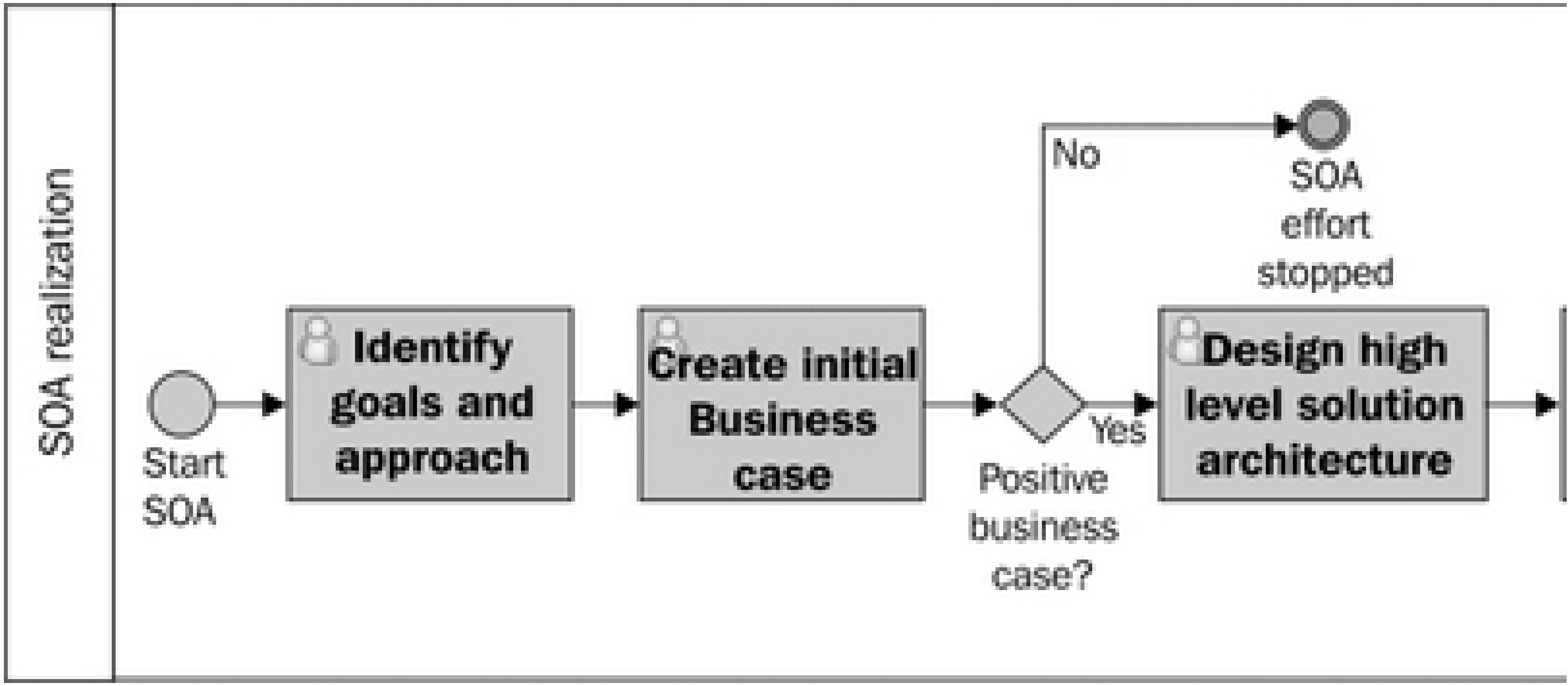
- MindReef – http://www.mindreef.com
- IBM – http://demos.dfw.ibm.com/on_demand/Demo/IBM_Demo_Rational_ClearQuest_Test_ManagementJun06.html?S=SWCA
- Mercury provides a service called ActiveTest that allows it to populate a Web service with real data loads from its server farm. Load runner has been enhanced to send same data to client and server and test for stresses.

Test tools

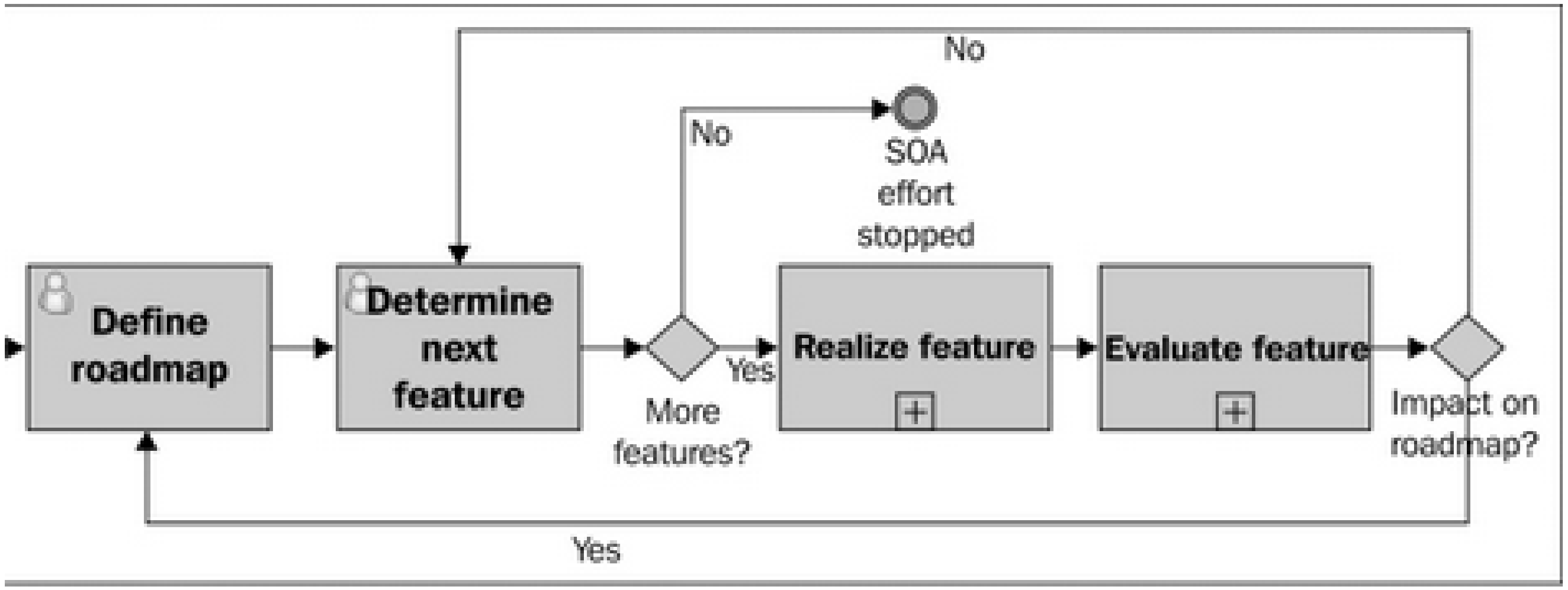
- Fiddler
  - HTTP traffic analyzer
- loadUI
  - Load testing – see soapUI
- WireShark
  - Packet analyzer, PuTTY

Project management

Organize

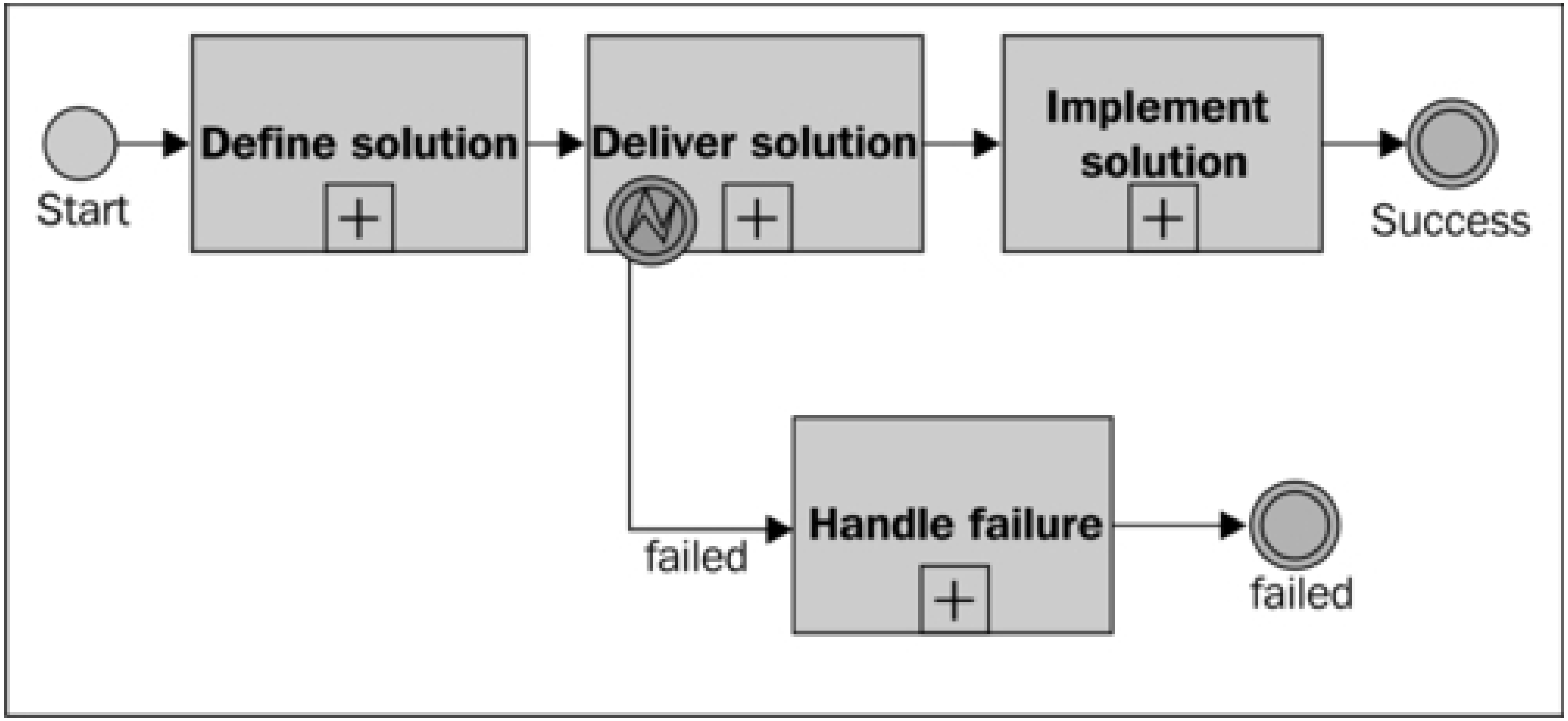

SOA realization
Start SOA
Identify goals and approach
Create initial Business case
Positive business case?
No
SOA effort stopped
Yes
Design high level solution architecture

Organize

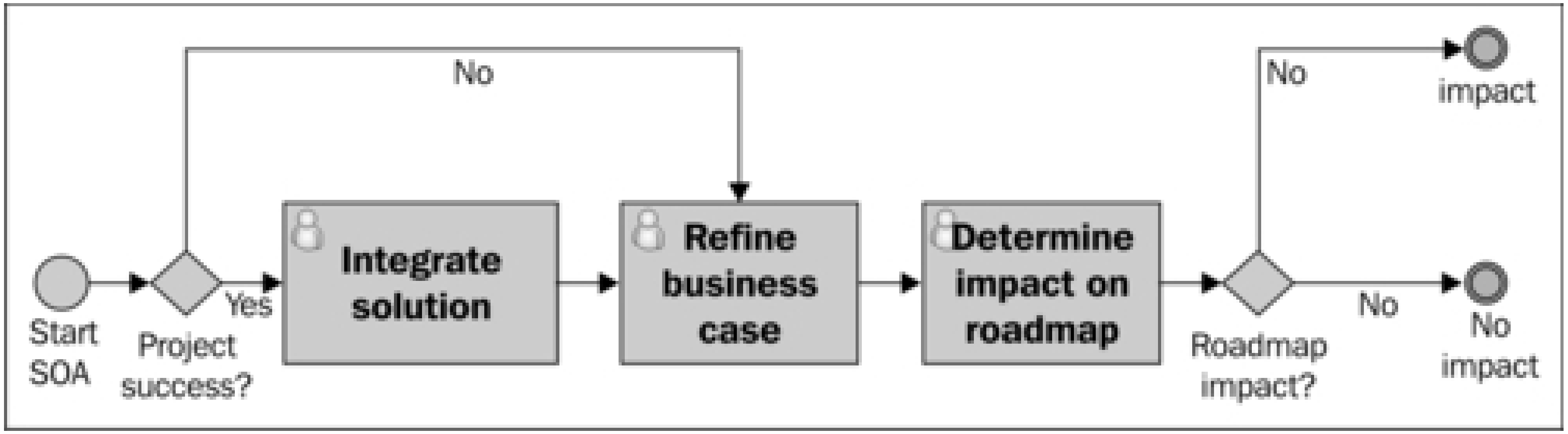


No
No
SOA effort stopped
Define roadmap
Determine next feature
More features?
Yes
Realize feature
Evaluate feature
Impact on roadmap?
Yes

Realize feature



Evaluate the feature



Business case

- Contents
  - Problem
    - What problem, affecting who, has what concrete bad consequences, that will have what good major effects when fixed?
    - No solution
  - Background
    - What type of problems do you face today, what is the history?
    - What business goals does the transformation contribute to?
    - What other projects are related to this transformation?
    - What trends and developments are relevant?

Business case

- Goal
  - Major effects quantified in time, money, etc.
- Future developments
- Constraints (specific requirements)
- General requirements, stakeholders
- Possible solutions
  - analysis of solutions
  - recommendation
  - next steps for solution

Roadmap

- Project management - work package identification
  - By business service - for IT driven projects
  - by process - business driven, business process reengineering (BPR) – service?, overloaded functions, groups of functions
  - by system, components - special cases
  - by feature - fast turnaround, easier with Agile – operations within the service, methods or functions

Maturity and stages

- A maturity model - pegging the tasks for a process by
  - low, medium, high skill level, informal to formal adherence
- SOA Level 0 - no experience
- SOA Level 1 – projects/processes locally rolled out - no reuse
- SOA Level 2 - projects/processes reused
- SOA Level 3 - projects/processes improved
- SOA Level 4 - projects/processes well managed
- SOA Level 5 - projects/processes optimized

Governance

- The allocation of decision rights to people who implement those decisions.
  - What decisions should be made is your choice
  - decide the balance of centralization and decentralization
- Business governance
  - the choices of what functional parts of the organization need to be created and the goals of those parts.
  - a business architecture

Change management

- The process that is used to evaluate and plan a feature to be added / changed / taken away.
  - small change - typically a bug
  - large change - enhancement or new project
- REST vs. SOAP example
  - aligns project goals with business goals
  - approved list of technologies?

Service lifecycle stages

- Identified - Analysis
- Designed
- Implemented
- Deprecated - tagged for retirement
- Retired

Versioning

- Version control is the management of changes to documents, software, and artifacts
- Codes
  - numbers, letters
  - revision suffix
- Versioning scheme
  - major vs. minor vs. update vs. patch

Types of change

- Contract
  - store hours
- Interface
  - FOH staff
- Implementation
  - manufacturers, BOH staff

Configuration management

- Tracking any change over time to anything
- Software configuration management
  - AKA version control
  - check in and check out controls
  - promotion and demotion to valid builds
- Release management
  - decides when software is ready
  - versions release

Major vs. minor changes

- Major
  - Change the state of the service to *deprecated* and determine the "end of life" date;
  - Create a new version of the service with the state *identified*
  - Announce the change to all current clients
  - Design the new version of the service and change the state of the new service to *designed*
  - Communicate the new design to clients who want to migrate

Major vs. minor changes

- Major
  - Realize the new version of the service and change the state to *implemented*
  - When the "end of life" date is reached, remove the old version
- Minor
  - Design the new minor version of the service
  - Announce the change to all current clients
  - Realize the new version of the service

Processes and standards

- The processes
  - Find a service
  - Troubleshooting
  - Communicate a change
  - Maintaining registry and repository
- Standards
  - UDDI
  - ebXML RegRep
  - Atom Publishing Protocol

Project tips

- Communicate drivers clearly.
- Train people. Don't start training too early.
- Communicate your successes.
- Reward people that adapt to the new way.
- Learn from the mistakes.
- Look for a sponsor.

End matter

Books

- ***Pattern-Oriented Software Architecture Volume 4: A Pattern Language for Distributed Computing*** by Frank Buschmann, Kevlin Henney, Douglas C. Schmidt
- ***Patterns of Enterprise Application Architecture***, Martin Fowler, David Rice (The Addison-Wesley Signature Series)
- ***Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*** (The Addison-Wesley Signature Series) by Gregor Hohpe, Bobby Woolf
- ***Remoting Patterns: Foundations of Enterprise, Internet and Realtime Distributed Object Middleware*** (Wiley Software Patterns Series) by Markus Voelter, Michael Kircher, Uwe Zdun
- ***SOA in Practice: The Art of Distributed System Design*** by Nicolai M. Josuttis, O'Reilly Aug 2007
- Thomas Erl SOA book series

Resources

- Magazines
  - SOA World is now Microservices Expo Blog
  - Service Technology Magazine Thomas Erl, editor
- CBDI
- SOA glossary
- Aurea Dr. K creator of BPM
- SAP customer stories

Resources - standards

- OASIS SOA committees
- W3C Technical Reports
- Original specifications

Resources

- Businesses providing web services (Data As A Service)
  - http://www.strikeiron.com/ - data validation
  - http://www.xignite.com/ - real-time financial data
  - http://www.serviceobjects.com/ - data validation
  - http://www.cdyne.com/ - data validation, demographic data
- Directory
  - http://xmethods.com