Java is a trademark of Sun Microsystems, Inc.

Java

JavaOne℠

SOA Solution Delivery

A case study using Agile delivery methods with SOA-RM

Mike Morris
Capgemini UK
mike.morris@capgemini.com

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Problem

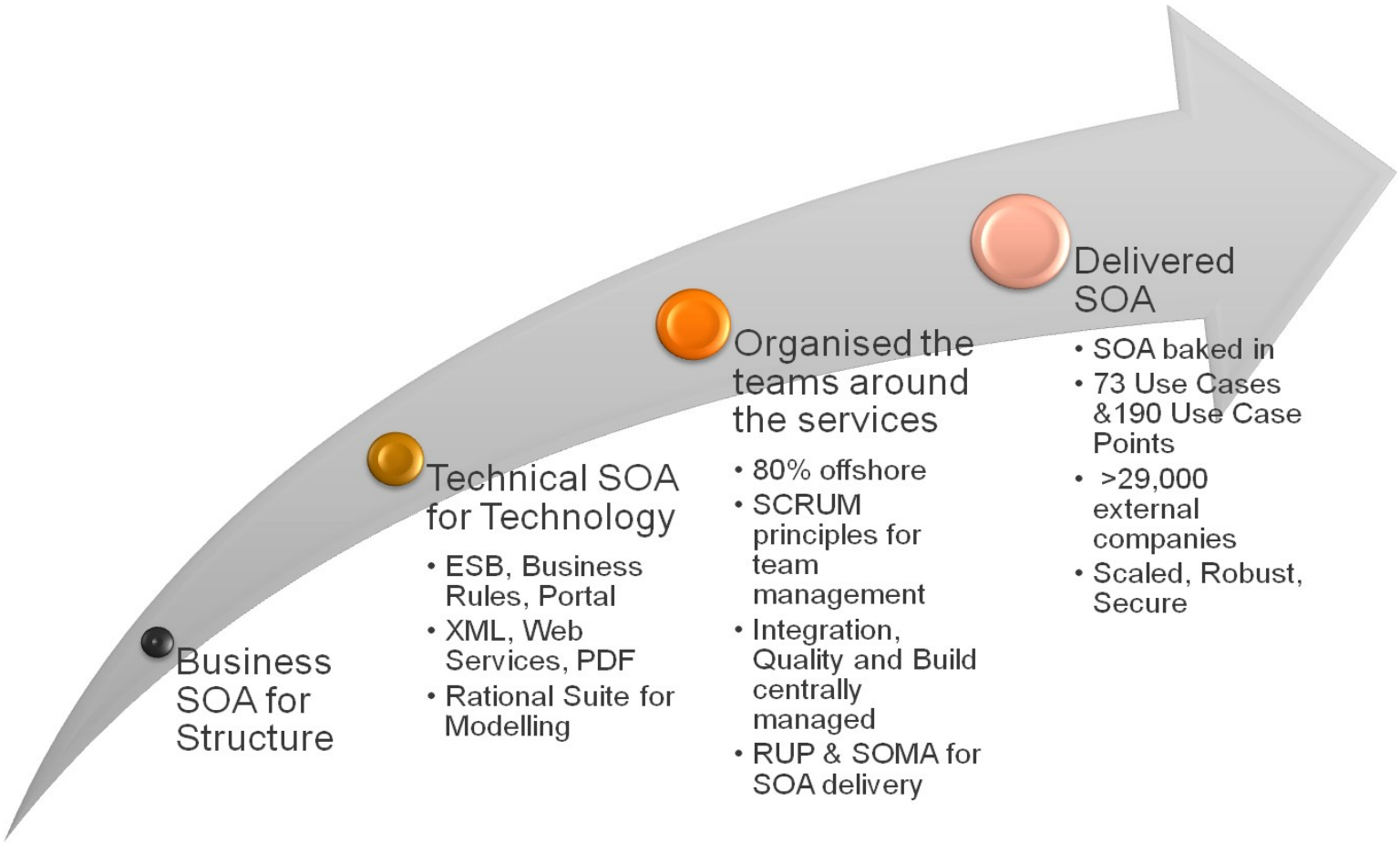
- You are a software development manager, software architect or development lead.
- Your project is about to go down the SOA route
- You spent much time selecting your Java™ technology stack, and
- you need to take advantage of multi-site resources (say for cost or experience reasons)
- Where should you begin?

The purpose of this talk is to look at a SOA design and delivery approach

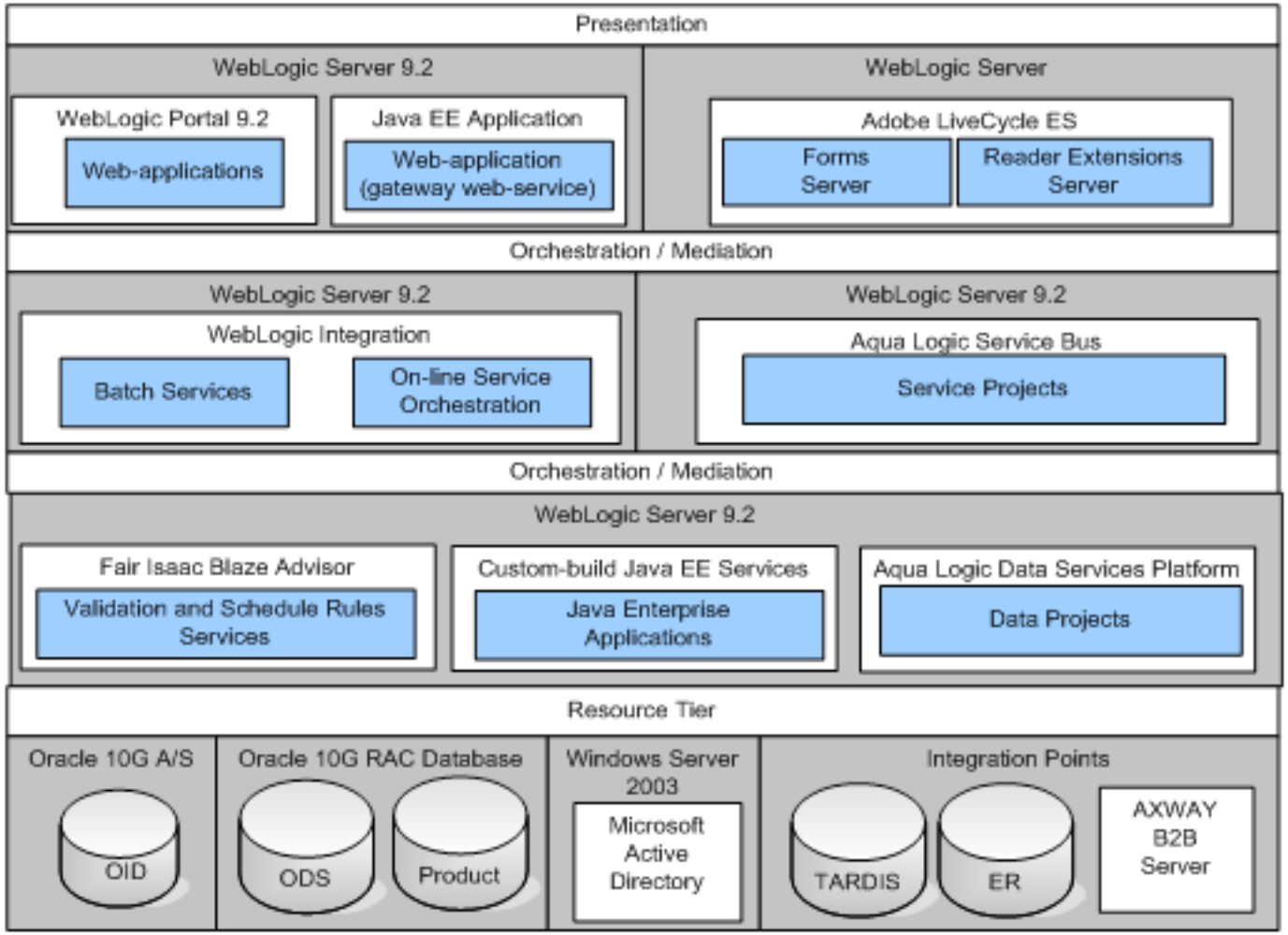
Financial Services Authority GABRIEL Project

- Goal
  - Build an Electronic Reporting solution to enable the FSA to move towards a risk-based approach to regulation
- Challenge
  - >29,000 regulated institutions
  - Complex schemas, new SOA technology stack, security and compliance
  - The need to enable future change and scalability
- Solution
  - Oracle, Adobe and Java-based middleware stack
  - Delivered end Q2-2008
  - EMEA Innovation award from Adobe
  - Happy customer

Highlight delivery approaches using the FSA GABRIEL SOA Journey



FSA GABRIEL SOA Solution Stack



A RELAXED LAYERED SOA SYSTEM

Agenda

- The right reasons
- Using the OASIS-RM to define services
- Using large-scale Scrum, agile practices and Open UP to manage the development process
- Design approaches and practices
- Other practices and Lessons learnt
- Recap
- Q&A

The right reasons?

- Do you have to do it this way?
  - Large, multi-site and offshore development => avoid
  - Complex SOA stack for a simple problem => avoid
- When should you do it this way?
  - You are building a software intensive system with 70-500 people
  - You are going to build many new systems to a common SOA architecture

SOA definition

OASIS-RM Recap

- "An Architectural Paradigm for organizing and distributed capabilities that may be under the control of different ownership domains.
- A framework for matching needs and capabilities.
- A view of architecture focusing on "Services" as a mechanism to allows interactions between those with needs and capabilities.
- A way of thinking about problems"

from TS-6887 (2008), Steve Jones and Duane Nichols

Partitioning methods and tools that enables the isolation, co-ordination and delivery of services by disparate development teams

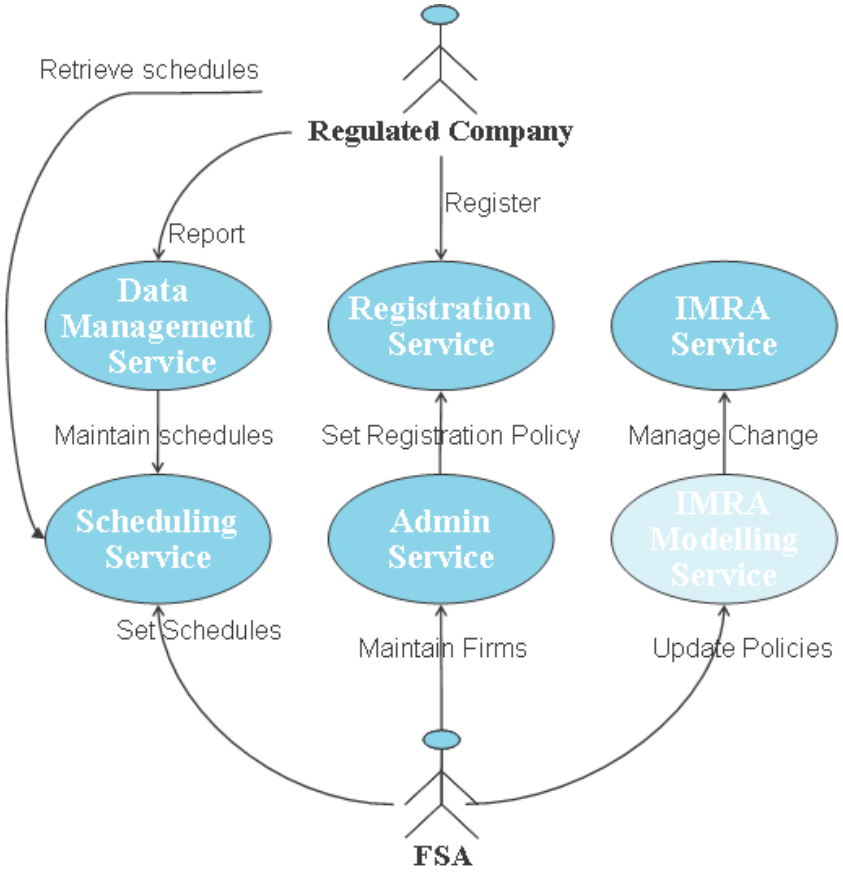
Standard Model for SOA
OASIS-RM Recap

- OASIS Reference Model gives us a standard abstract model for:
  - Service
  - Service Description
  - Capability
  - Visibility
  - Execution Context
  - Policy
- Ensures we have a ubiquitous standard language for discussing services which isn't domain-specific

Business Service Modeling

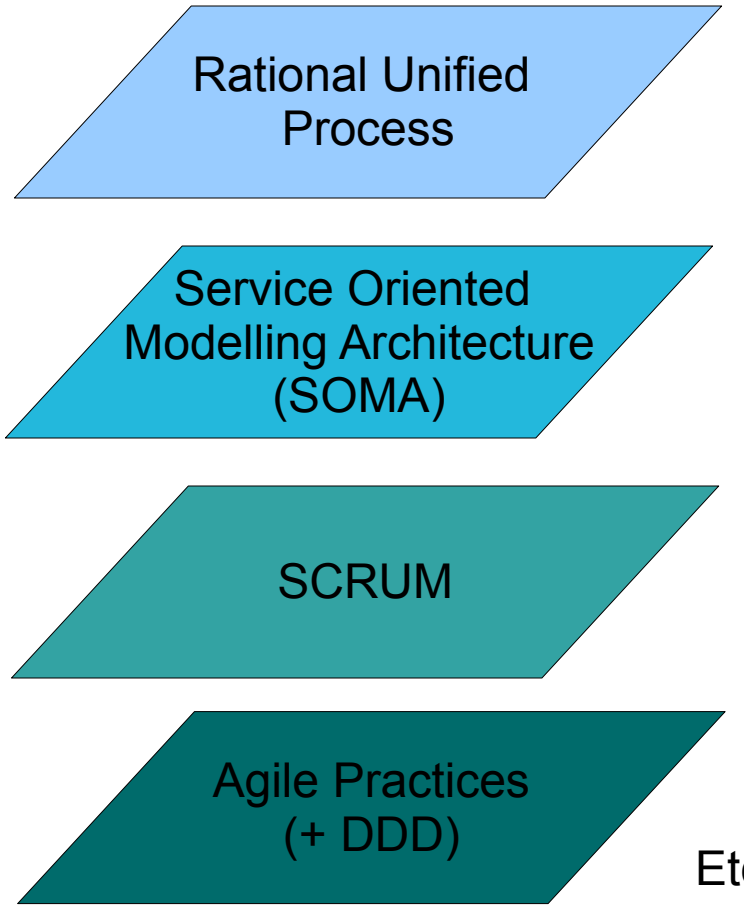
- Modeling services with the business gives us alignment between business and development teams
  - Output → Business Service Model
  - Duration → 1-2 day workshop
  - When → Early before your dev work starts
  - Why → Used to identify projects, begin to form a ubiquitous language and Isolate the domain model into BOUNDED CONTEXTS.

Example service architecture



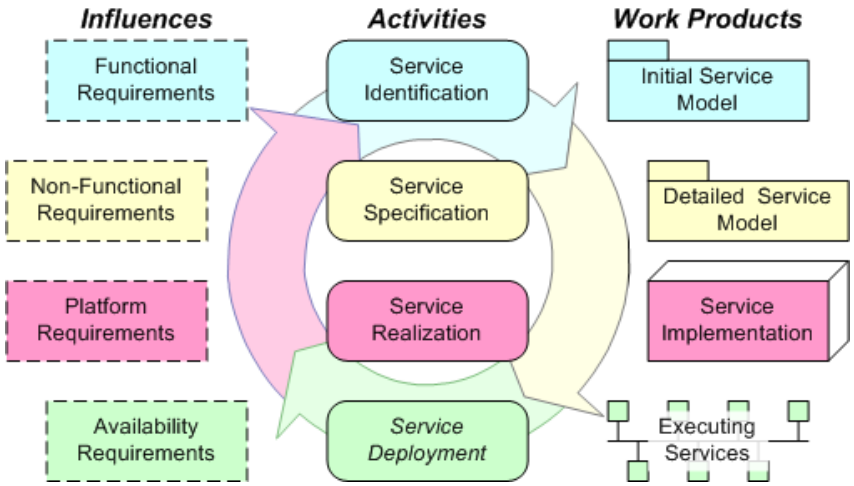
- Define the service providers
- Define the high-level interface contracts
- Break the problem down
- Quick and simple
- Partitioning decisions made early
- Drives team and technology organisation

SOA Development Life-cycles



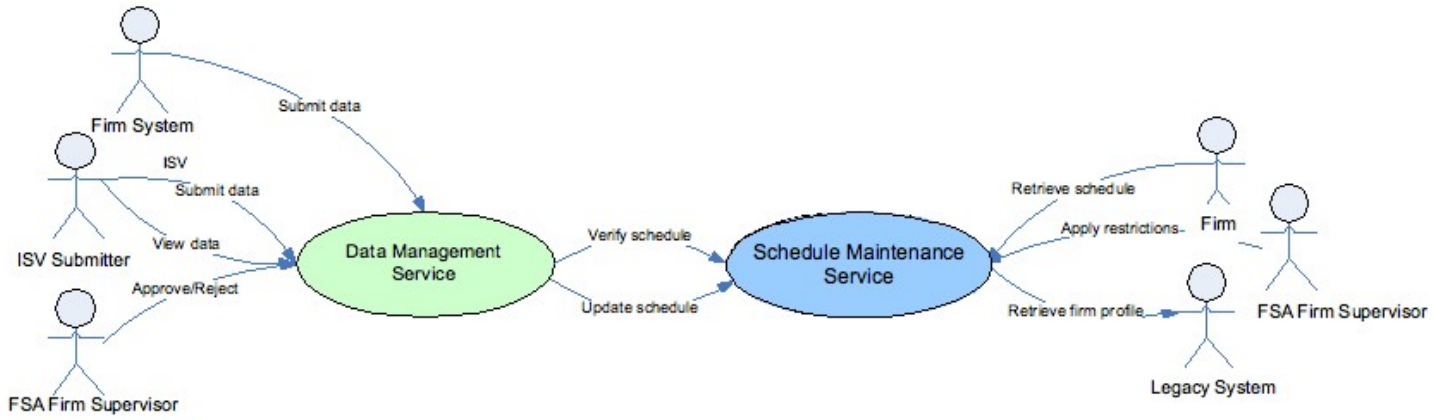
- No one size fits all
- Most methods could work
- We used
  - SCRUM
  - Rational Unified Process with SOMA
- Do not forget the "domain model" → DDD

RUP SOMA for Large-scale systems



- Apply Services Litmus Tests
- Service Specification
- Message Design
- Identify Security Patterns
- Subsystem Design (SOA)
- Component Specification (SOA)

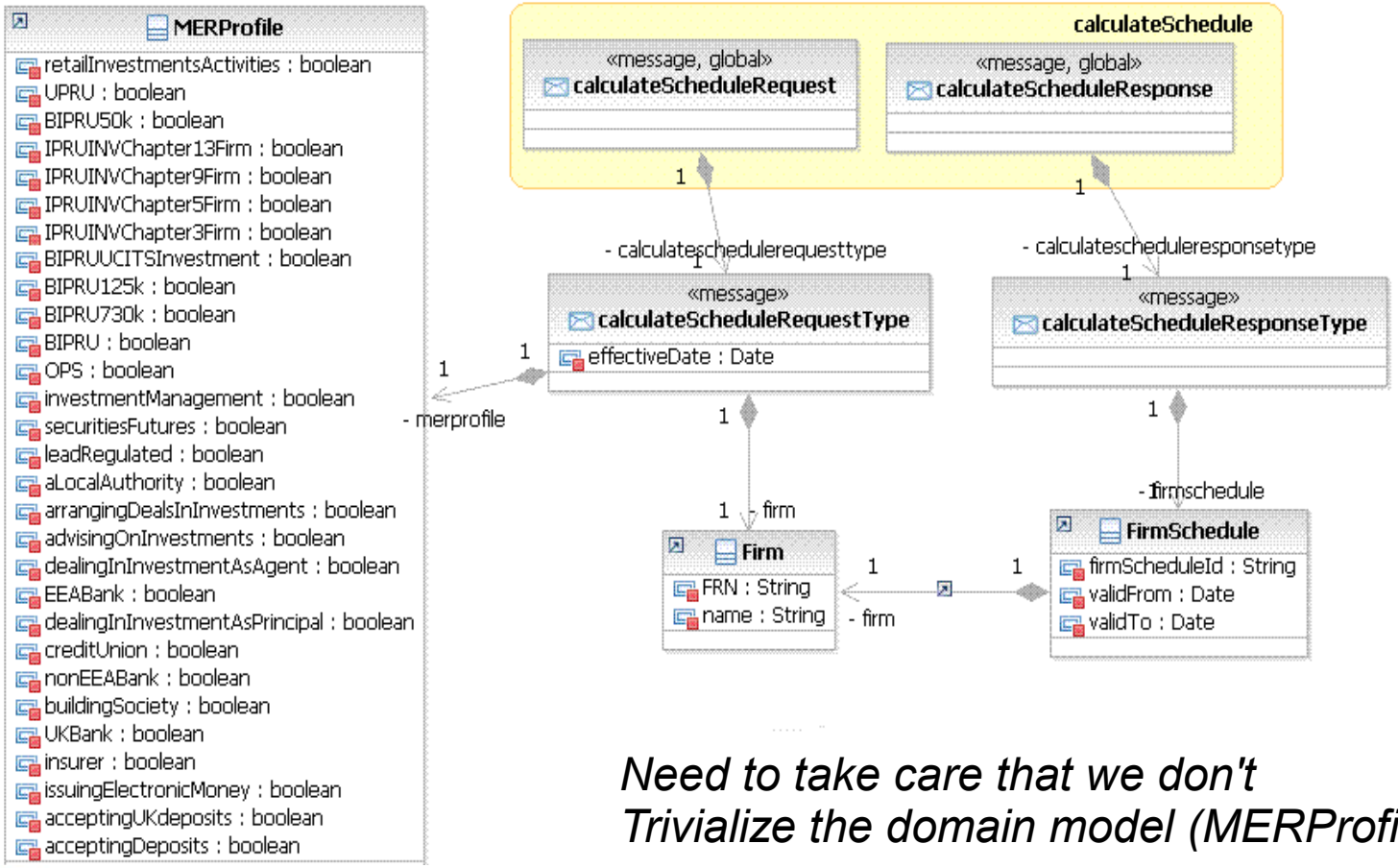
Service model example (1)
Service model notation



WHO? WHAT? WHY?

Service Model Example (2)

Defining a formal service specification based on messages



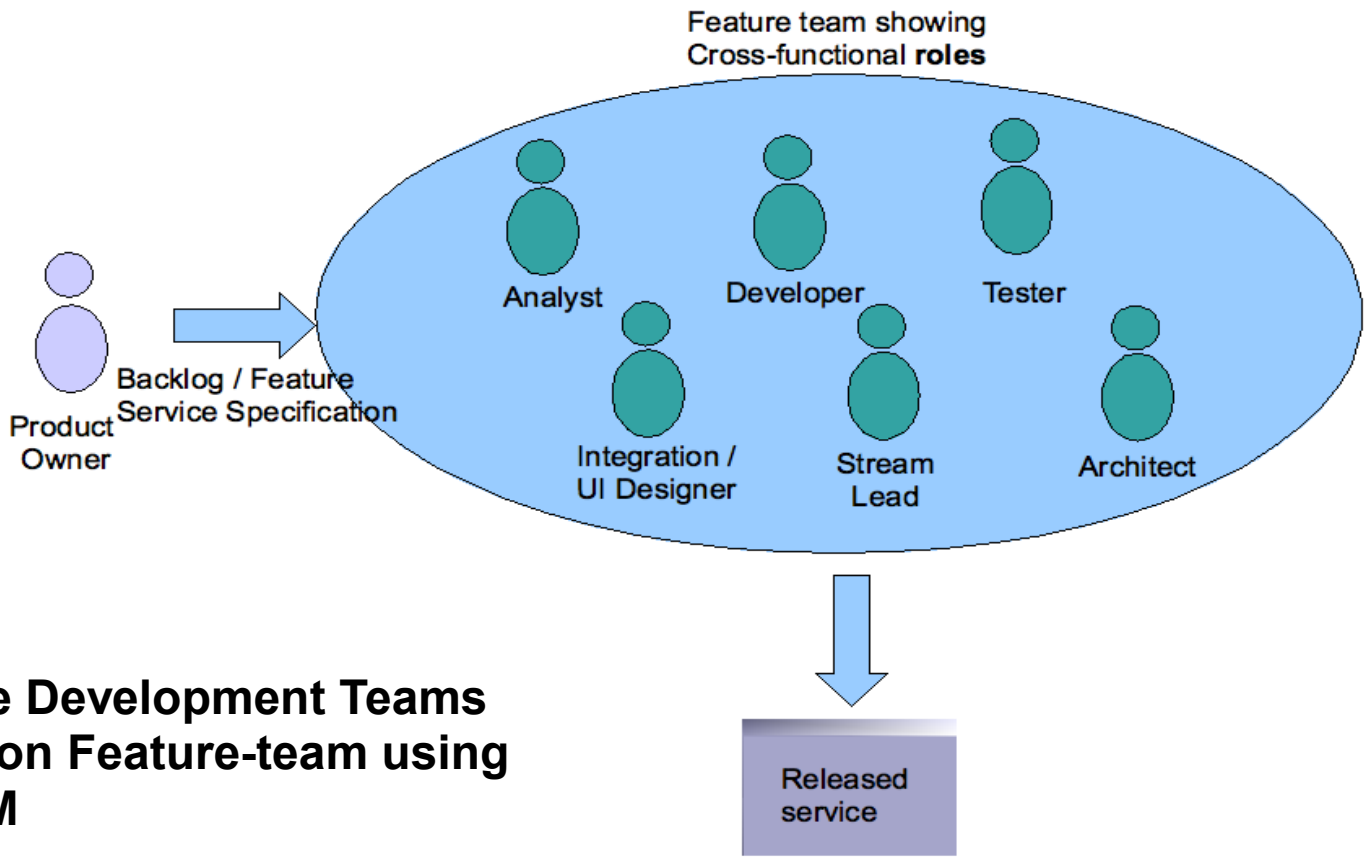
Need to take care that we don't
Trivialize the domain model (MERProfile)

Service Model Example (3)

Defining a formal service specification based on messages

- Focus on the domain model
  - Scenarios
  - Unit Tests + Code
  - Iterate
- Be driven by the contract
  - Pre-condition, Post-condition, Invariants
- Keep a watch out
  - leaky abstractions
  - interface mis-use ('it looks the same')

SOA Team organisation tools
Based on feature teams (1)



Service Development Teams based on Feature-team using SCRUM

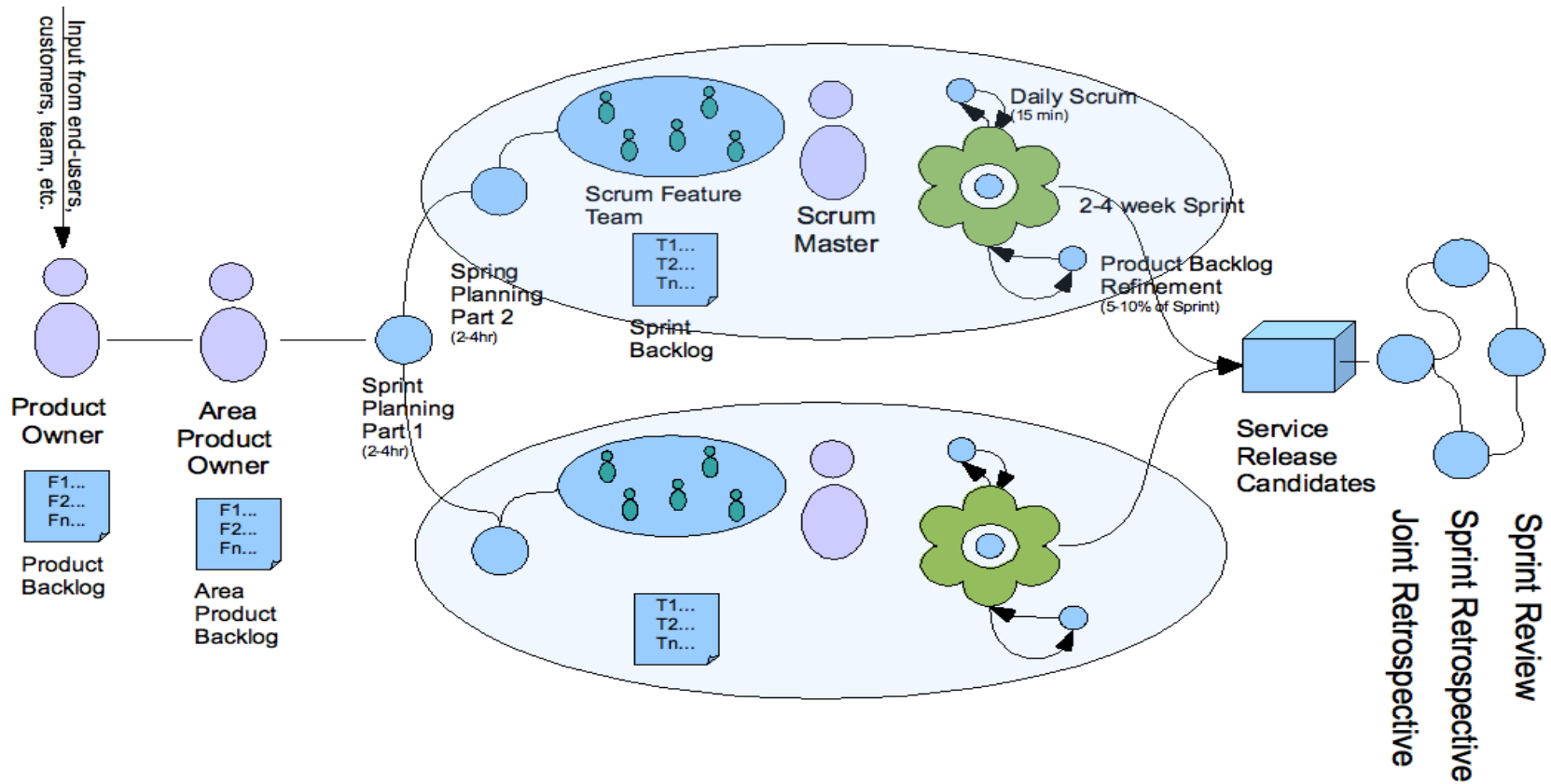
SOA Team organisation tools
Based on feature teams (2)

- Teams (either physical or virtual) centered on a set of project services within a business service domain or Context
  - long-lived
- Co-located for each service
- Each team comprises 7 ±2 people
- Cross-functional and cross-component
  - “vertical teams focussed on business functionality” [Jutta Eckstein]
- Generalizing specialists

SOA Team organisation tools
Based on feature teams (3)

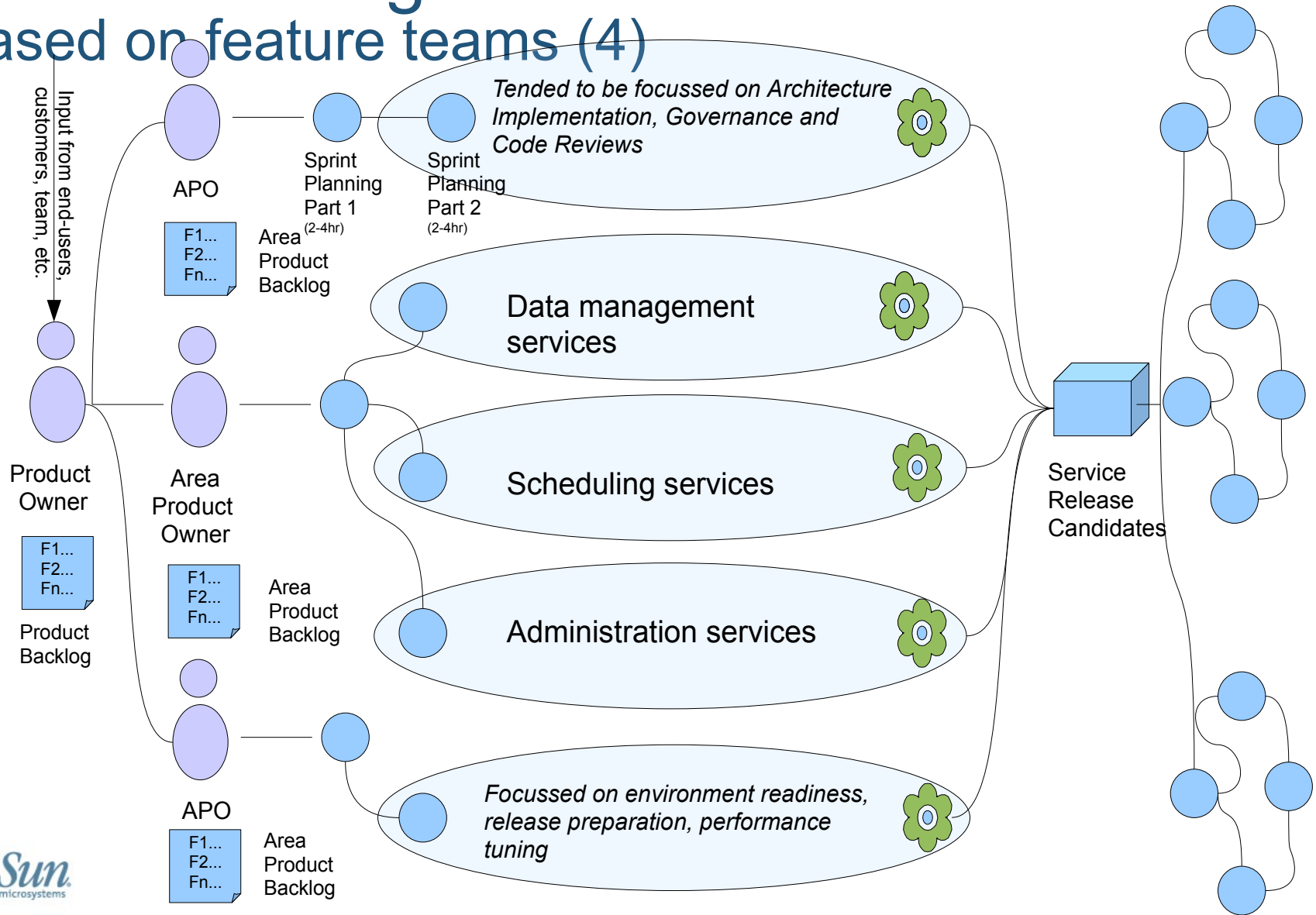
- Empowerment
- Accountability
- Measurement
- Identity
- Consensus
- Balance and diversity

SOA Team organisation tools
Based on feature teams (4)



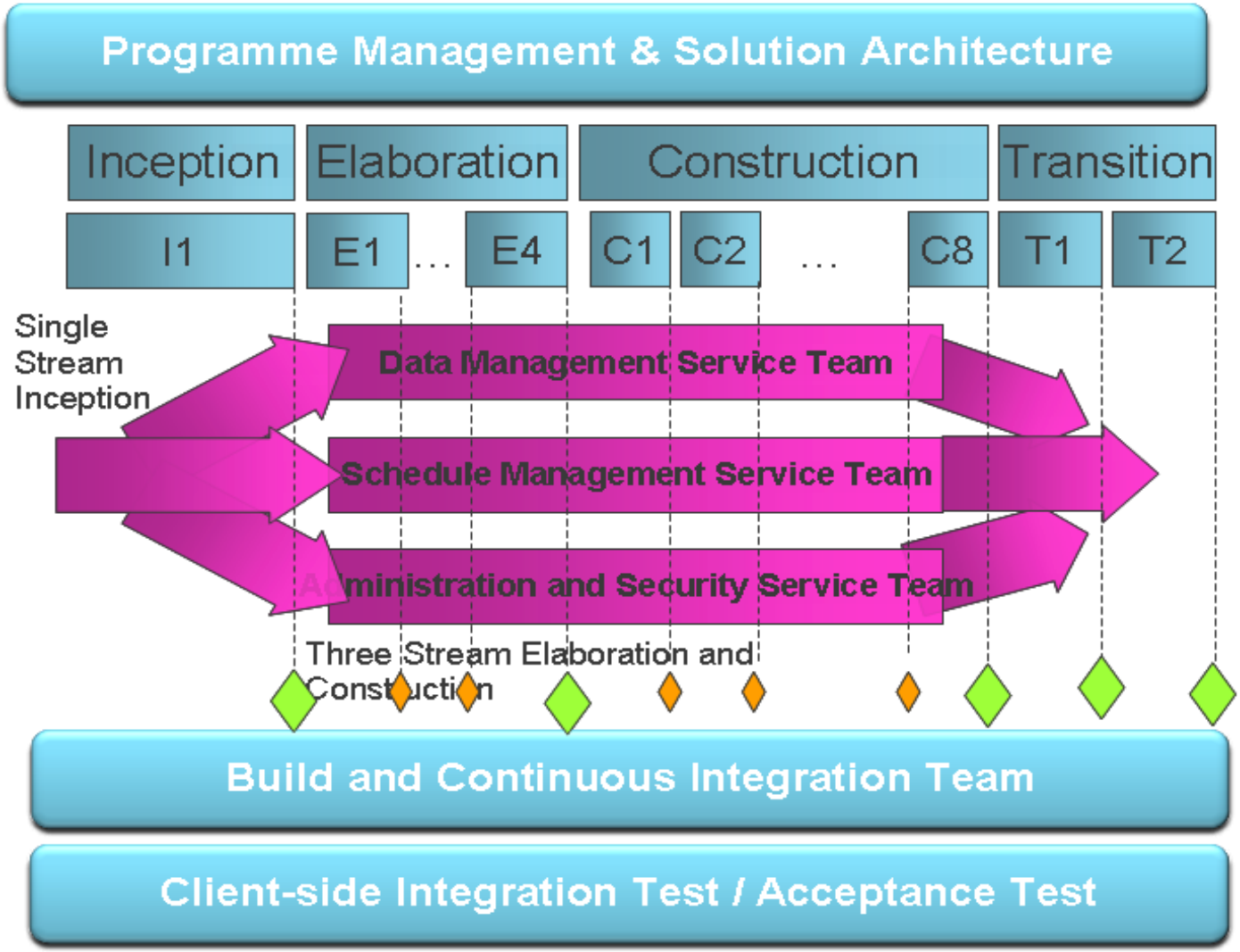
SOA Team organisation tools

Based on feature teams (4)



FSA GABRIEL Project Organisation

Using RUP and SCRUM practices for Service Teams

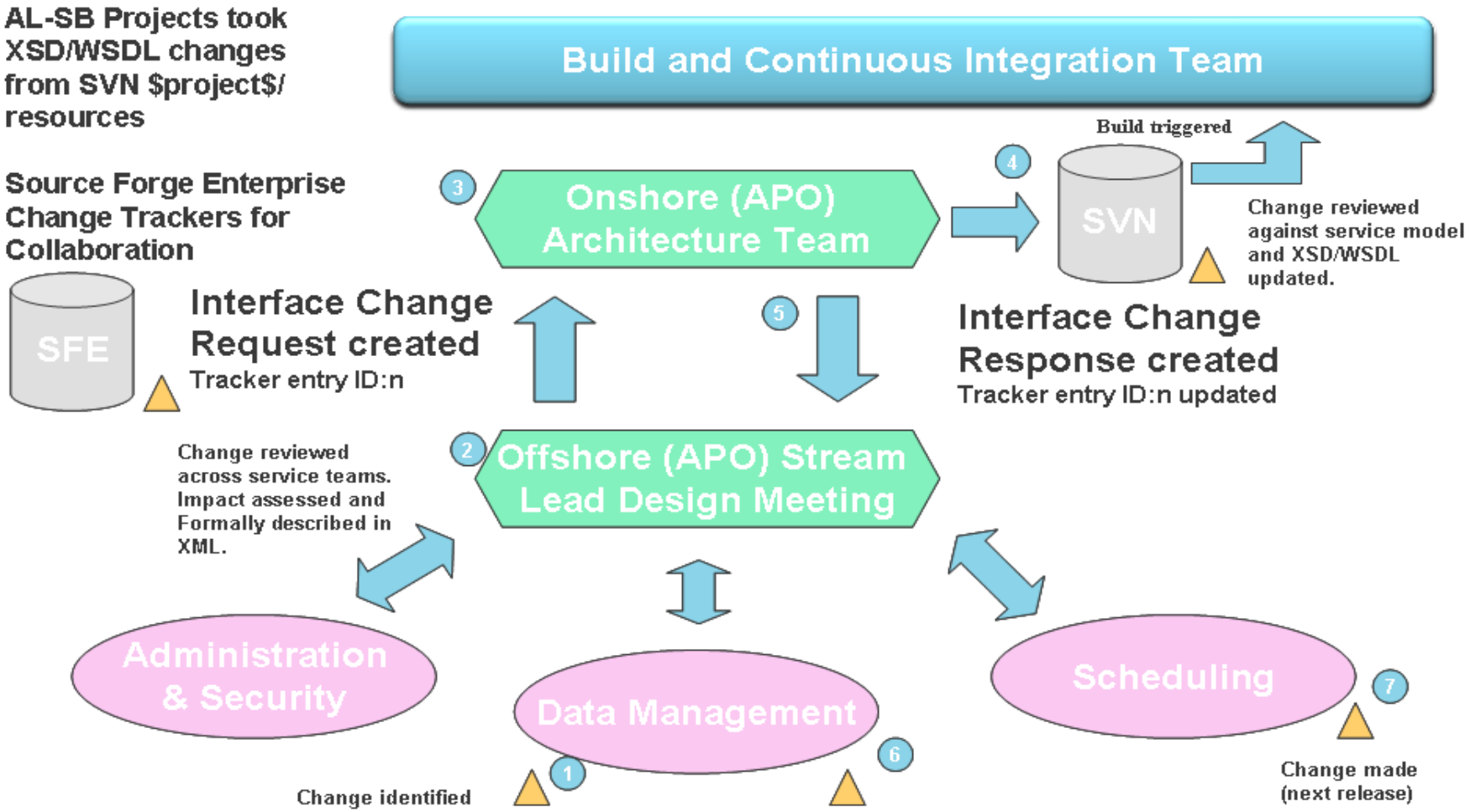


SOA Team organisation tools
Enhancements - Interface Management

- Often there will be 2 types of service consumers
  - Internal - Service development teams
  - External – The regulated companies
- Interface definition and control was critical
  - External interfaces had to be as near as 'right first time as possible'
  - Manage and communicate changes using tools
- Strong interfaces enables service teams to be loosely coupled
  - Reduces communication

Interface Management

Needed onshore review of interface changes



Configuration Management

- Structure the Version Control System by Service
- Use dependency management tools
  - e.g. Ivy to manage libraries, XSDs and WSDLs
- Do Continuous Integration
  - Invest in scripting releases and build of runtime containers (domains, clusters, etc).
- Setup defect management system
  - Integrate to your Version Control Tool
- Define a service provider runtime versioning scheme

Lightweight governance practices

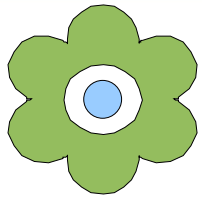
- "The essence of [the Toyota system] is that each individual employee is given the opportunity to find problems in his own way of working, to solve them and to make improvements."

  Wakamatsu and Kondo, Toyota

Using services to organise technology

- Source code and defect management
  - Modules, SVN, ParentDefectID projects
- Portal projects
  - Using WSRP for partitioning
- Oracle Service Bus
  - Definition of projects and Business Services
- Oracle Data Services Platform
- Rule services
- Structuring orchestration projects

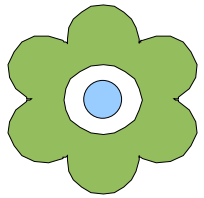
Services, a Software Process AND Design



- Inextricably linked
- Use the service model as a starting point to establish contracts
- But break it when you learn more (i.e. Knowledge Crunching)
- Avoid choosing technology that forces your domain model to go a certain way
- We need to look at which design approaches fit complex SOA delivery
  - Domain Driven Design versus Smart UI

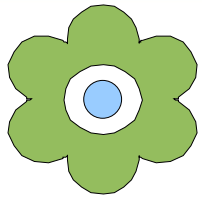
Domain driven design and SOA Impl

DDD has a lot to offer SOA implementation



- Prevent accidental syntactic re-use of object attributes from a service interface
- Focuses on the domain-model to define a UBIQUITOUS LANGUAGE
- Provides a construct for SERVICE to encapsulate application services or domain services
- Provides constructs for strategic design choices which impact the domain, the service model and the development teams
- But....in Domain Driven Design you don't start with service definition

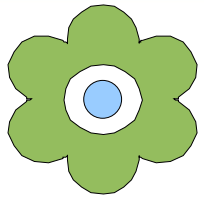
Isolating the domain versus defining behind an isolated boundary



- In other words, top-down versus bottom-up
- A DDD approach would typically identify application layer functions from the model:
  - these could be implemented as services or perhaps just classes
- Isolate using the service model (top-down)
  - but use a DDD approach as the refinement of the model
- Over time the business service model will become less used by developers and more used by management

Domain-Driven Design - Concepts

Concepts - Enriching the model



- Building Blocks
  - Entities
  - Value Objects
  - Aggregates
  - Services
  - Repositories
  - Modules
  - Factories
- Supple Design
  - Intention-revealing interfaces
  - Side-effect free functions
  - Assertions
  - Standalone classes
  - Conceptual contours

Services to Domain Objects
and vice versa

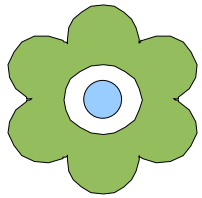
- Service Model: Dealership Level

Using the domain model to refine Services v. service decomposition

- Domain Object: Car

Using scenarios to drive out the domain model

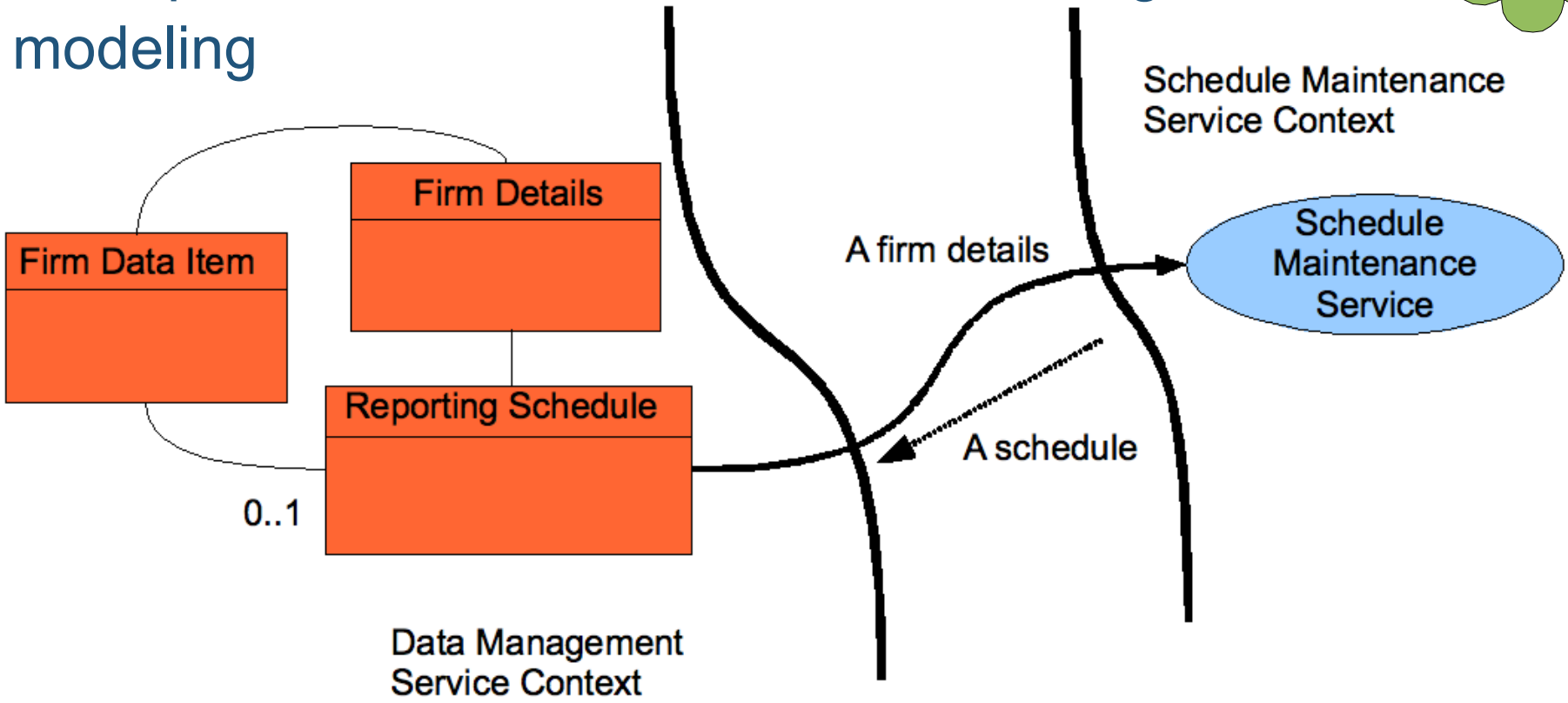
Domain-Driven Design
Concepts - Enriching the model



- Patterns
  - Bounded Context
  - Context Map
  - Customer/Supplier
  - Shared kernal
  - Conformist
  - Anti-corruption layer
  - Separate Ways
- Techniques
  - Abstract core / Segregating the CORE
- Large-scale structures
  - Evolving order
  - System metaphor
  - Responsibility layers
  - Knowledge level
  - Pluggable component framework

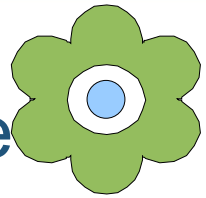
Bounded context and domain modeling

Example: Realize the service model through domain modeling

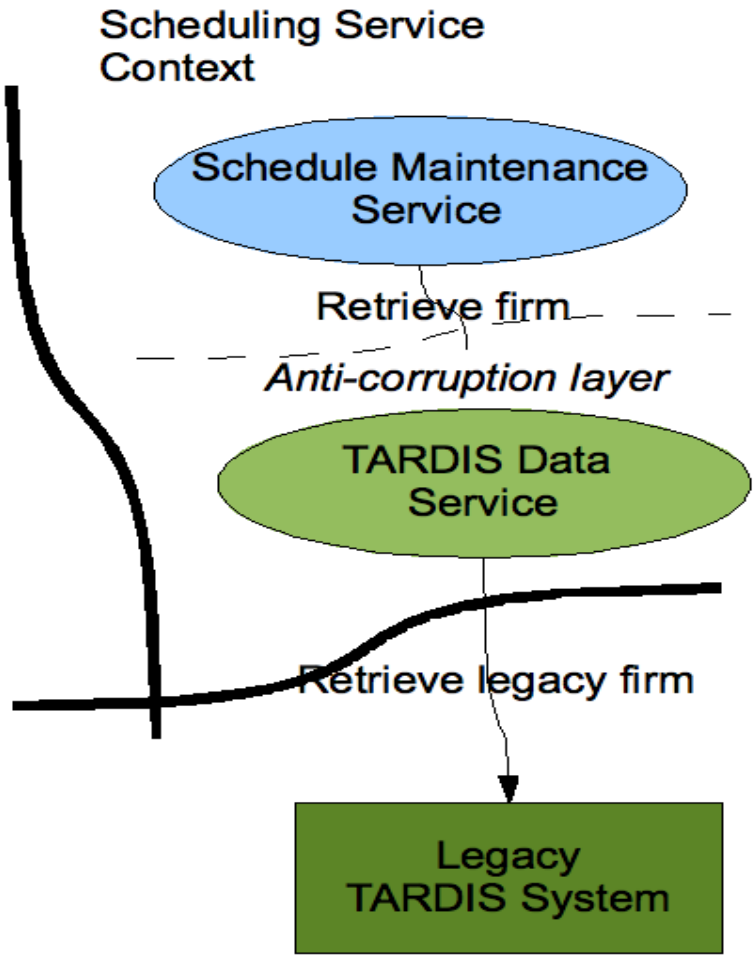


Anti-corruption Layer

Using a Service Bus and Data Services to provide model isolation



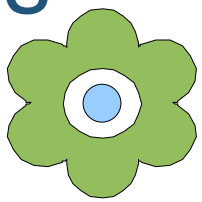
- Motive: avoid a client/supplier relationship with legacy
  - Conformist considered but insufficient access to dev team
- Anti-corruption layer isolates the two models (ours and theirs)
- ACL Implementation
  - Oracle Service Bus and
  - Oracle Data Services Platform



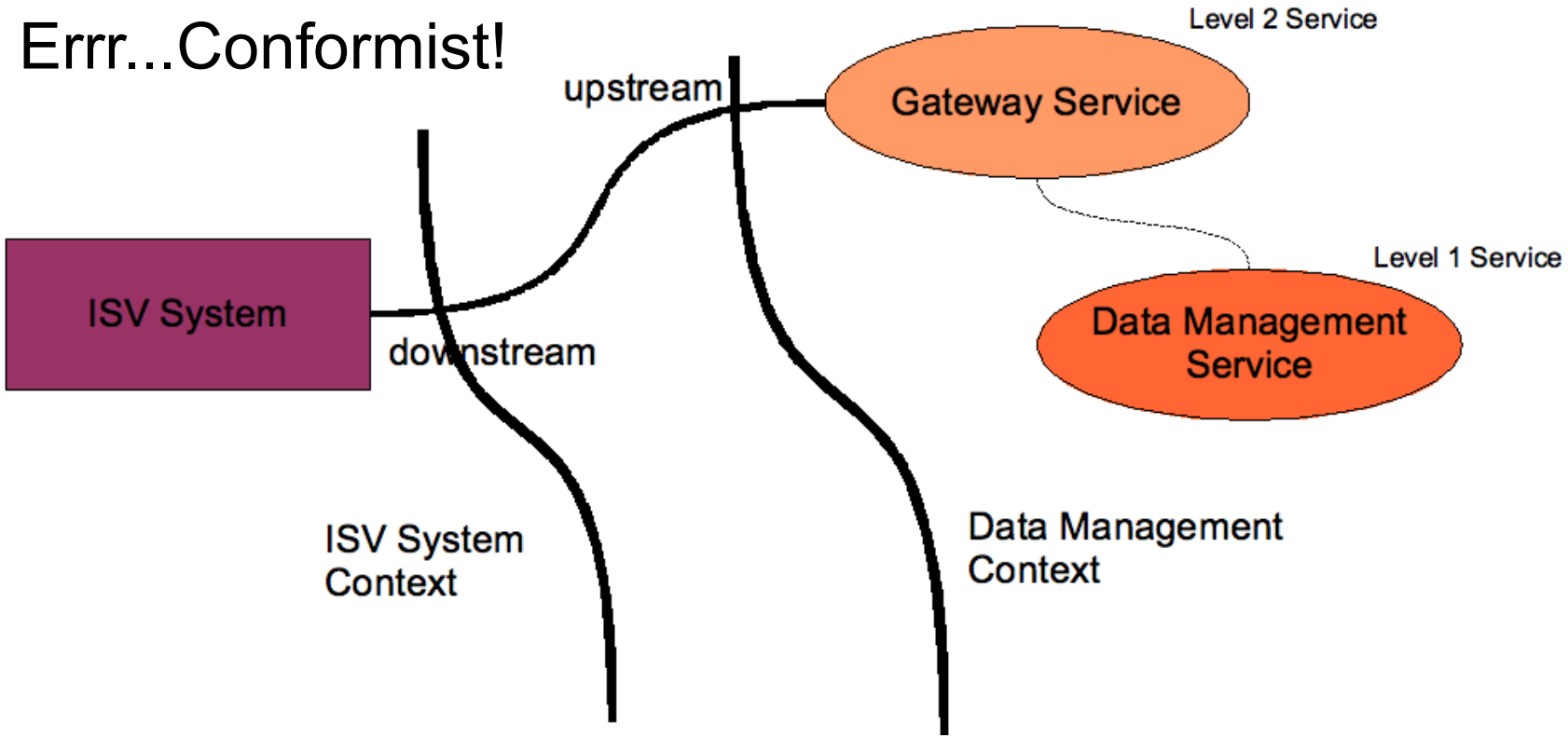
Big Ball of Mud

Customer/Supplier Development Teams

Our relationship to Firms and ISVs



- We mandated a web-service interface to other Firms and ISVs
  - Errr...Conformist!



Other project lessons learnt

- Don't be afraid to kill your butterflies
- Don't forget other standard practices, e.g.
  - TDD, Refactoring, Continuous Integration, Mocking
- Your organisation must be setup for agile development
- Ensure service operations act on a single Entity
- Beware of generality and favour a supple design
- Slow ramp-up during early iterations and little steps
  - Allow time for *foundation components*

Re-cap

- Service modelling
- Scrum/UP for Large-scale distributed projects
- Aligning technology setup to Services
- Best practices around agile development, Interface Management and configuration mgmt.
- Using domain-modelling for complex problem domains
- Lessons learnt
- Do you need the complexity?

Offshore SOA projects require more agility not less

References

- Web:
  - http://www.domaindrivendesign.org
  - http://www.fsa.gov.uk/Pages/Doing/Regulated/Returns/IRR/mer/tech_pack/index.shtml
- Print:
  - Mythical Man Month, Fred Brooks
  - Domain Driven Design, Eric Evans
  - Scaling Lean & Agile Development, Craig Larman and Bas Vodde
  - Enterprise SOA Adoption Strategies, Steve Jones

JavaOne℠

Thank You

Mike Morris
Capgemini UK
mike.morris@capgemini.com
http://developer-greenhouse.blogspot.com/