Testing of SOA based systems

Staged acceptance
Chris C. Schotanus

Experience the commitment®

Introduction

CHRIS SCHOTANUS

Principal IT Consultant

REQUIREMENTS & QUALITY MANAGEMENT

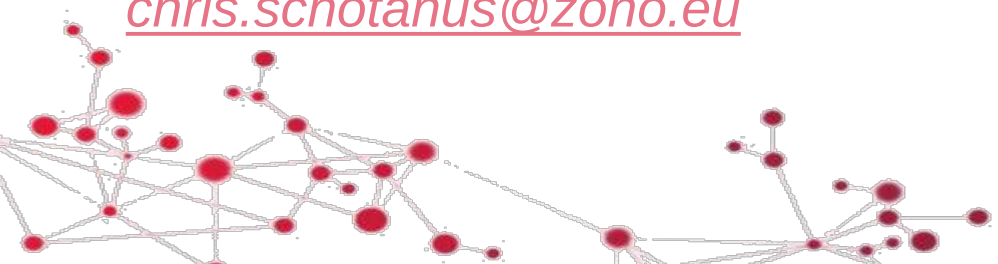
41 YEARS OF IT EXPERIENCE
OF WHICH 34 WITH CGI (AND ITS ANCESTORS)

27 YEARS IN TEST AUTOMATION, TEST PROCESS IMPROVEMENT & PROCESS ARCHITECTURE

Experience the commitment®

chris.schotanus@cgi.com

chris.schotanus@zoho.eu



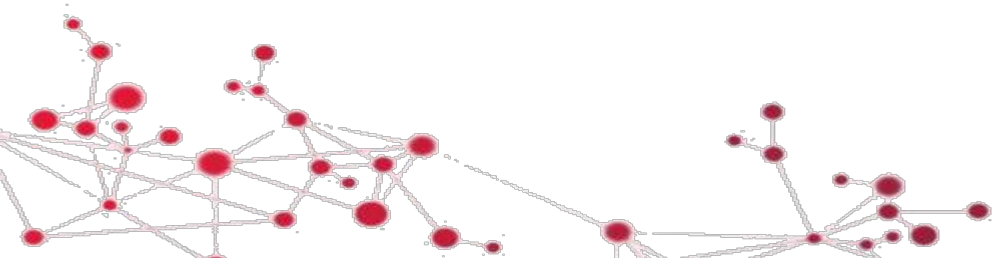
Today’s Agenda

01 Introducing SOA

02 Test specific items

03 SOA governance aspects

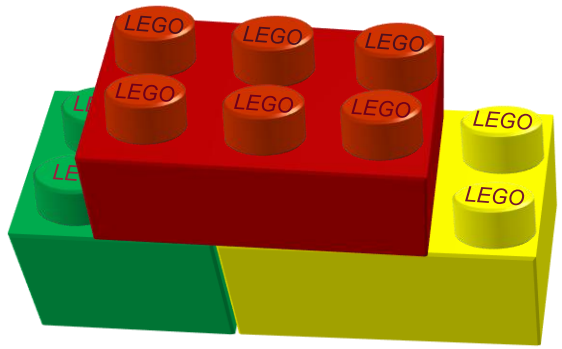
04 The real life experience



SOA based systems defined

(Ideal) SOA based systems are composite systems made of parts that are

- **Virtual**: the consumer is not aware of the implementation of the service
- **Standardised**: there is only one implementation of a function or responsibility
- **Modular** (replaceable) and compositional
- **Abstract**: generic, not dedicated to one specific consumer
- **Detached**: consumer and supplier are fully independent of each others implementation
- **Agnostic**: there is no direct link or relation between services

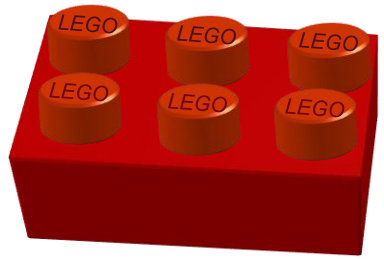


These parts are called "services"

A service is a bundle of resources that belong to the same context and that are able to solve operational problems and to support processes. Services are characterised by autonomy, loose coupling, reusability and a service contract, which abstracts the service implementation (technology and logic) via a well-defined service interface.

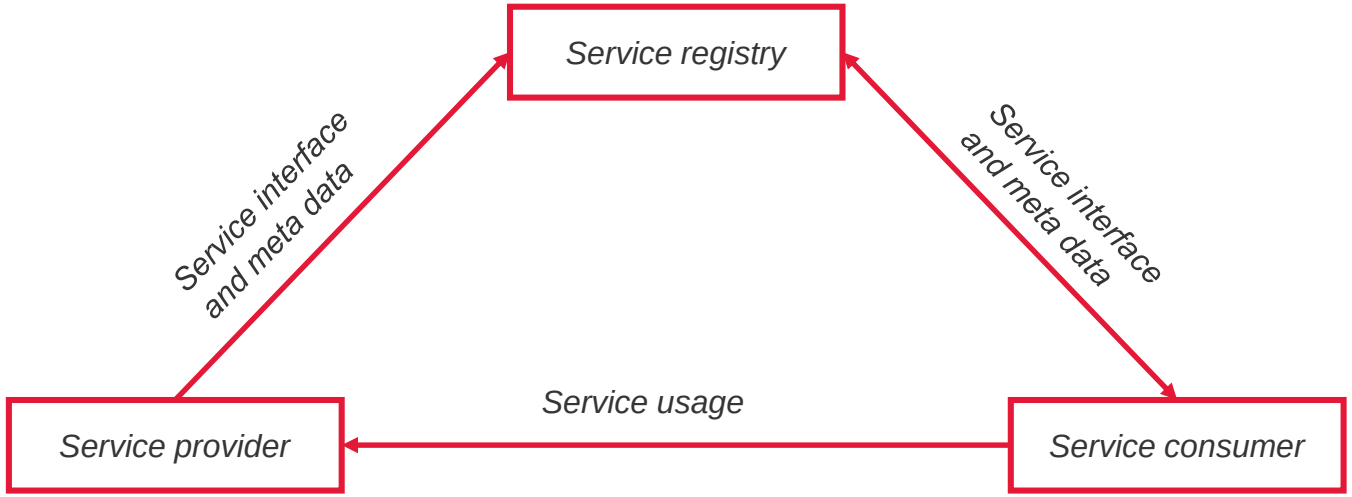
Types of service contracts:

- **Basic** contract: service operations incl. input/output parameters
- **Behavioural** contract: parameter combinations, pre/post conditions, service call sequences
- **Synchronisation** contract: simultaneously service calls
- **QoS** contract: non-functional requirements



SOA principals: service triangle

A service call involves the service provider, the service consumer and the service registry. All information needed to perform a service call can be gathered during run-time („service binding“) which enables SOA to allocate resources flexibly and dynamically.



SOA principals: service composition

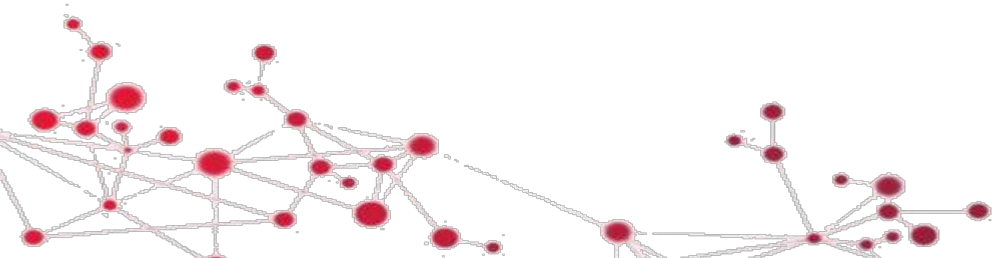
Services can be categorised according to their functional stage of expansion:

- **Basic** services
- **Composed** services, using compositions of other services to perform a complex action
- **Process** services, representing long running business processes

Service composition can be achieved in two ways:

- **Orchestration** (service calls are coordinated by a central instance)
- **Choreography** (self-governed peer-to-peer communication between services)

Ideally, SOA implementations provide sophisticated workflow engines to allow configuration of user-defined process definitions.



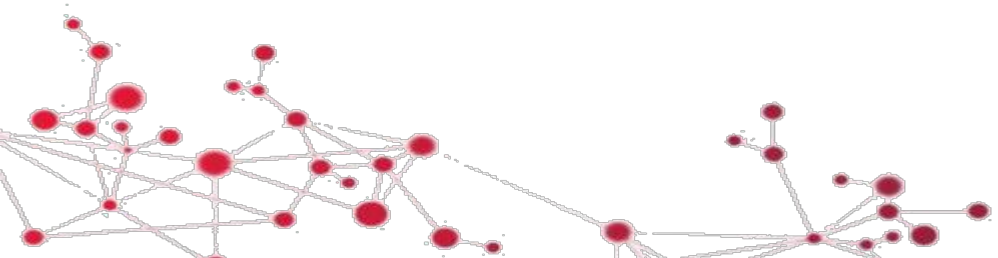
CGI

SOA principals: service mediation

When a service call is performed, the code of the service implementation is not integrated into the service consumer‘s environment but remains and is run in the service provider‘s environment. The interoperability issues which result from such an approach are addressed by a mediation system (e.g. ESB). In order to support complex data exchange scenarios and business process, SOA is typically based on messages.

Examples of message exchange patterns:

- One way (Fire and Forget)
- Request/Response
- Request/Call-back
- Publish/Subscribe



What this presentation is about

Questions often asked regarding SOA are:

- Who will accept a services?
- How about ownership of services?
- What should we test if one or more services change?
- Interface testing versus end-to-end integration testing?
- Who owns the maintenance budget?

These are the questions I will try to answer

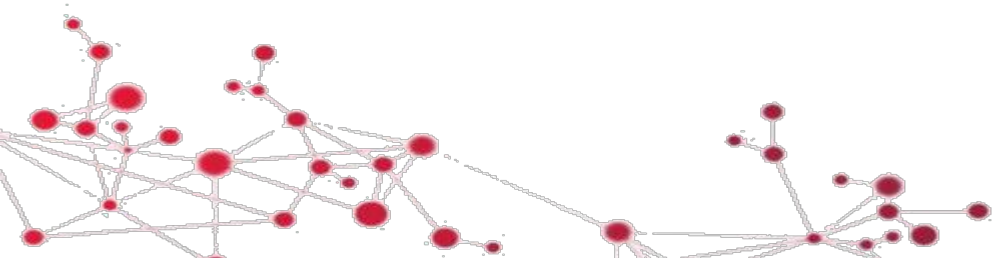
Today’s Agenda

01 Introducing SOA

02 Test specific items

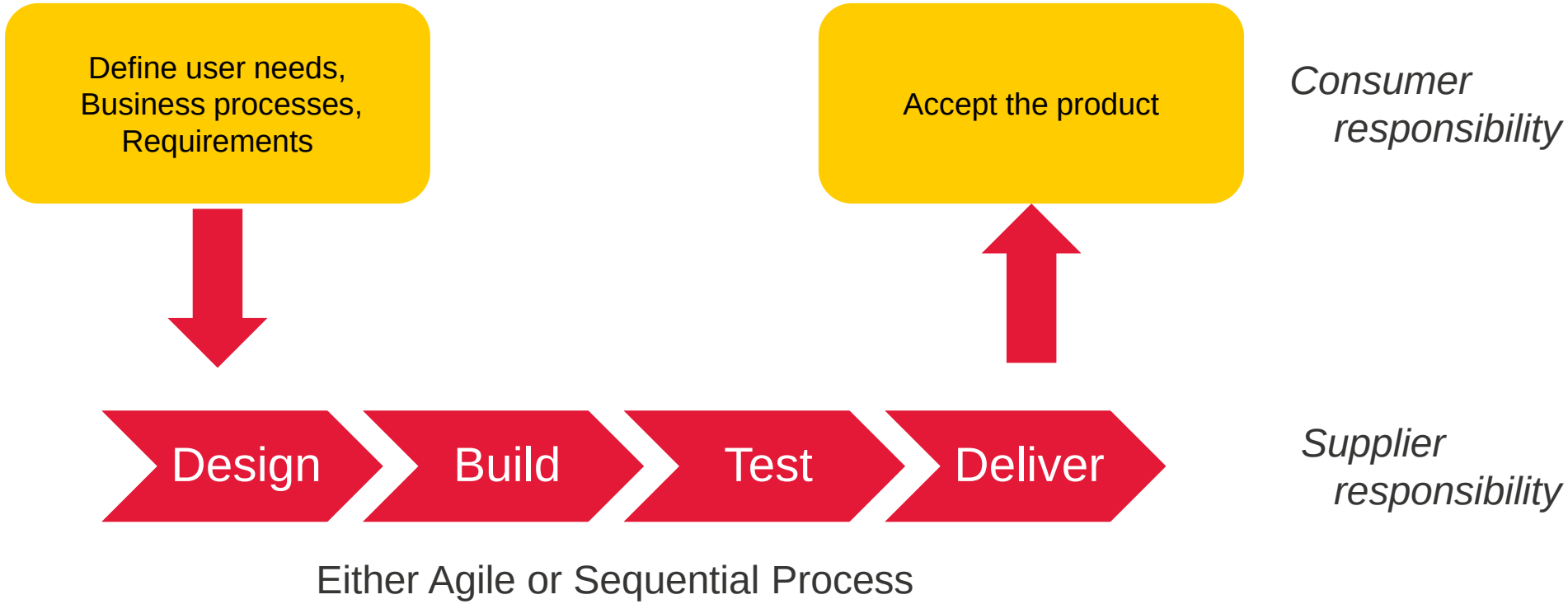
03 SOA governance aspects

04 The real life experience



Testing services

The supplier/consumer model as reference

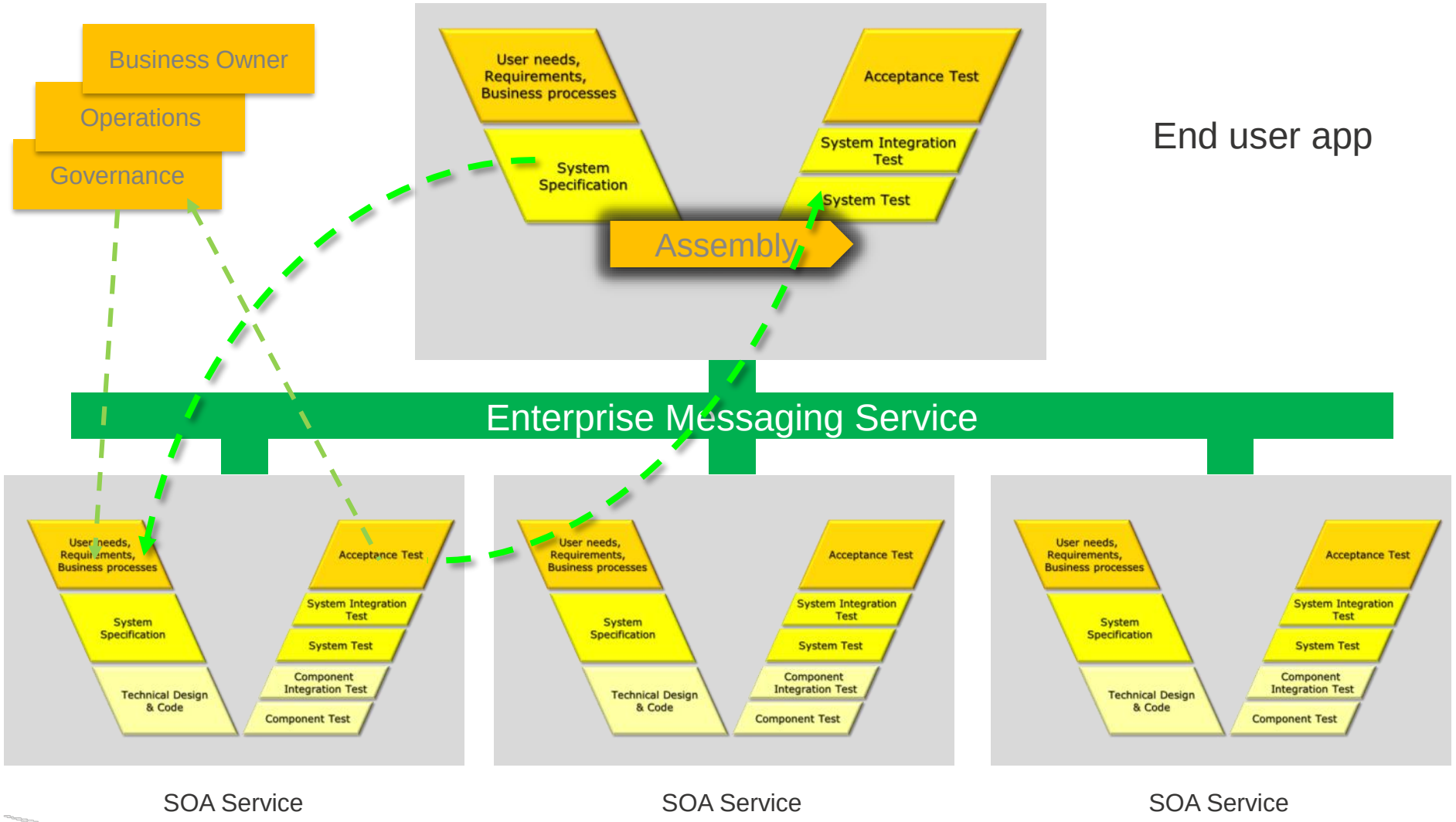


The SOA acceptance process

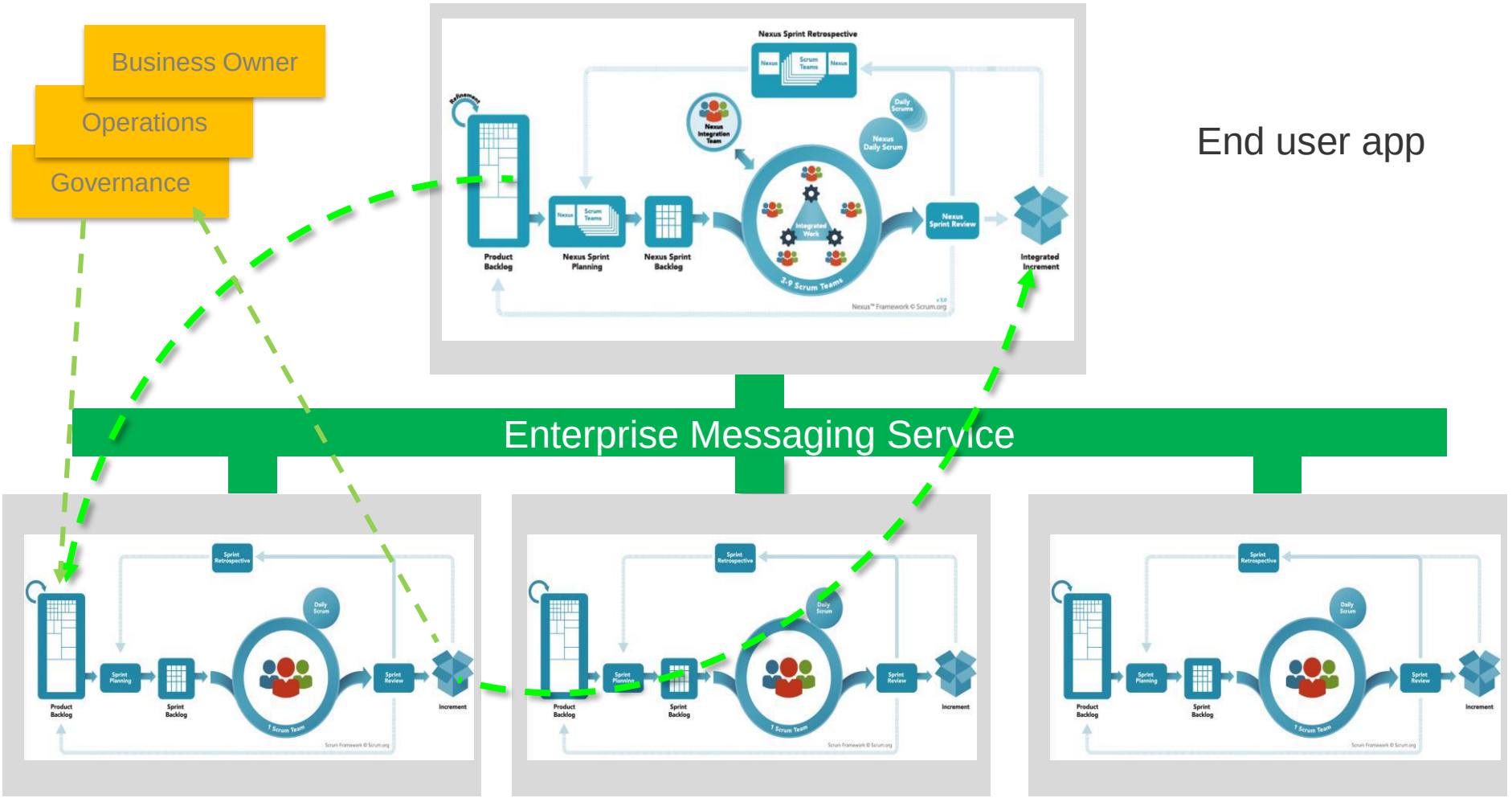
- User applications are assembled from or use (web)services
- Service requirements are derived from business requirements
- Services are selected from service registry
- Systems are assembled using services

It is a Staged Acceptance Process

The SOA specific test process



This might as well be a hybrid or a full agile process like Nexus



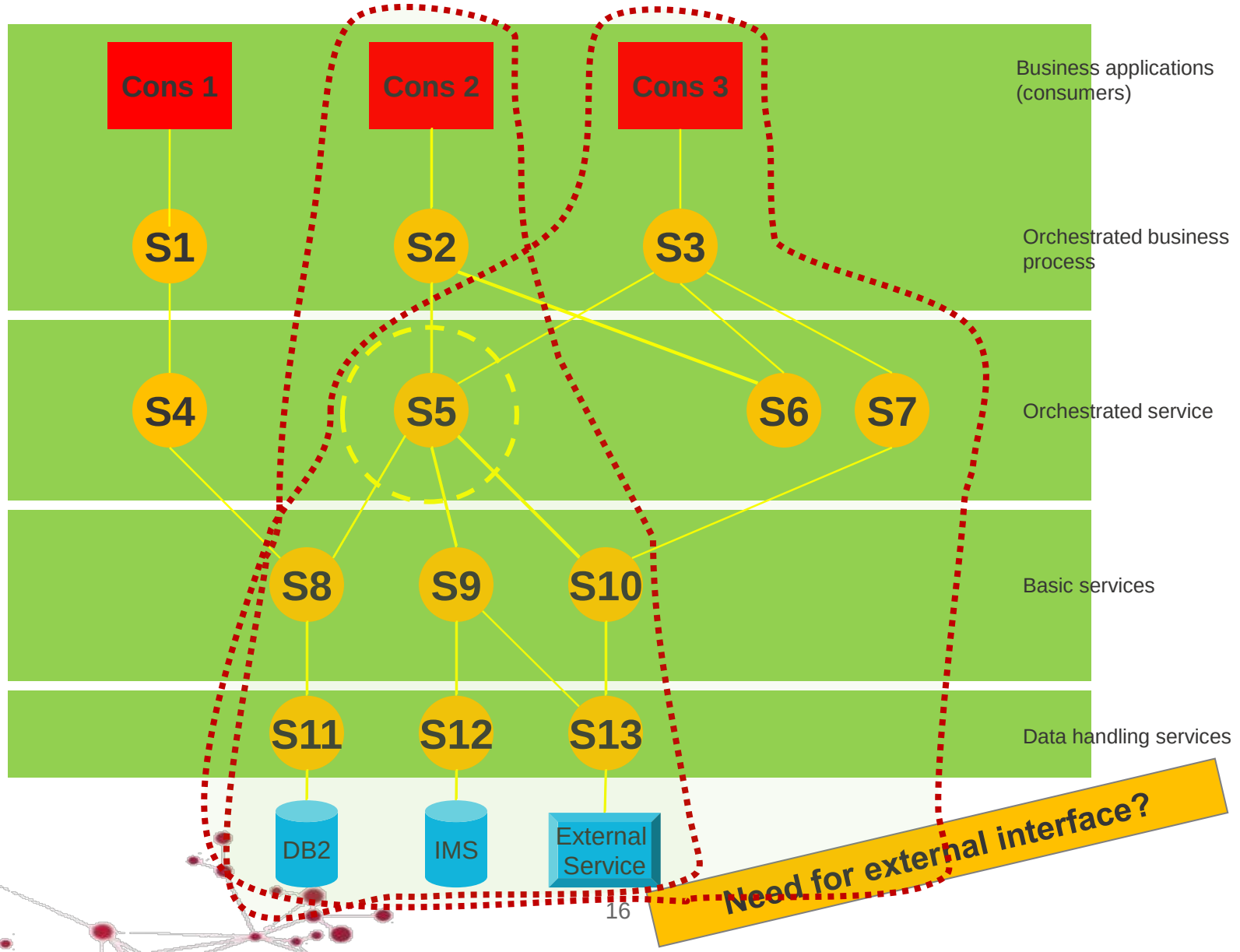
End user app

Service Integration (Regression) Testing

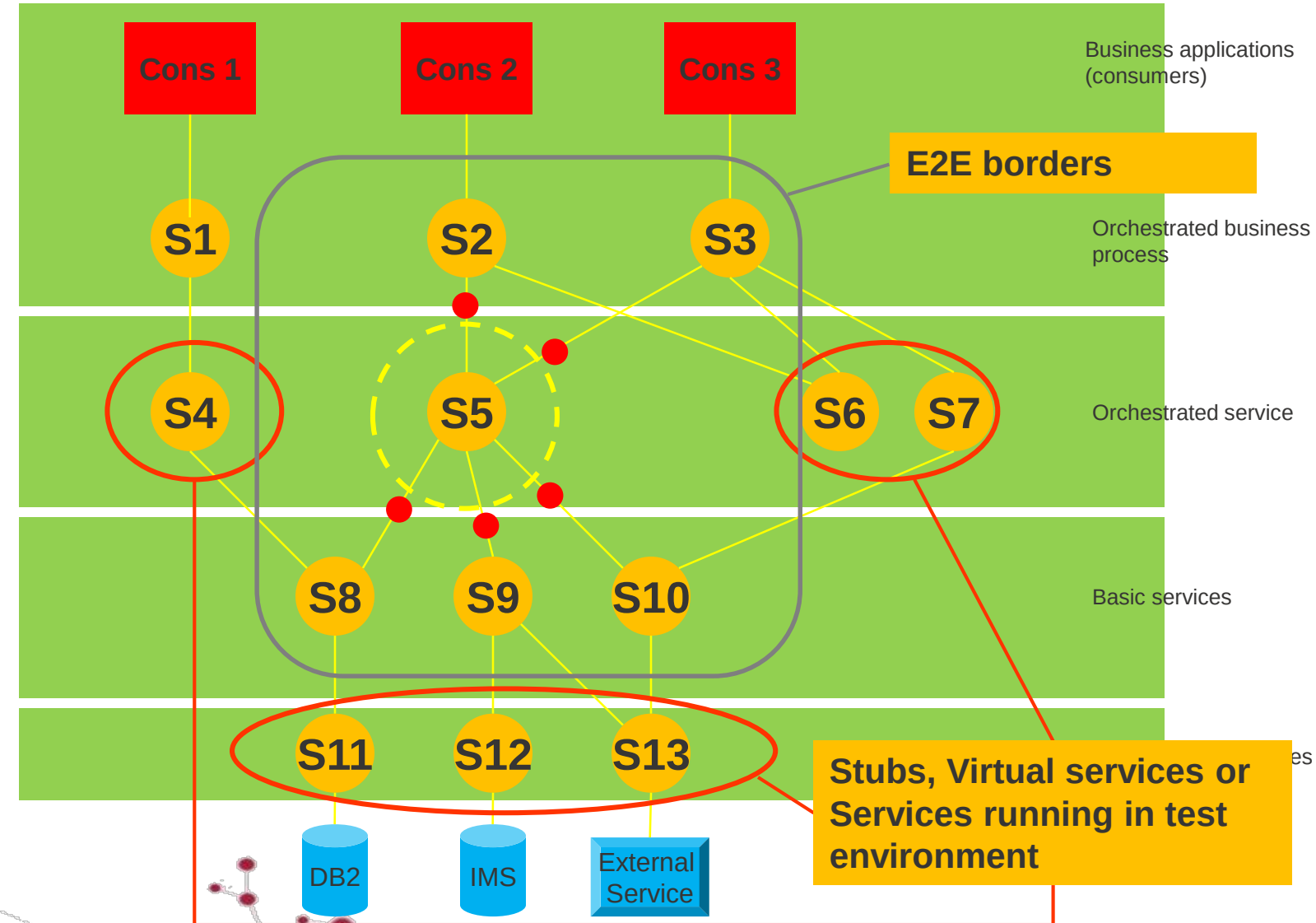
What should we test if a service is changed?

End-to-End testing versus
Interface testing

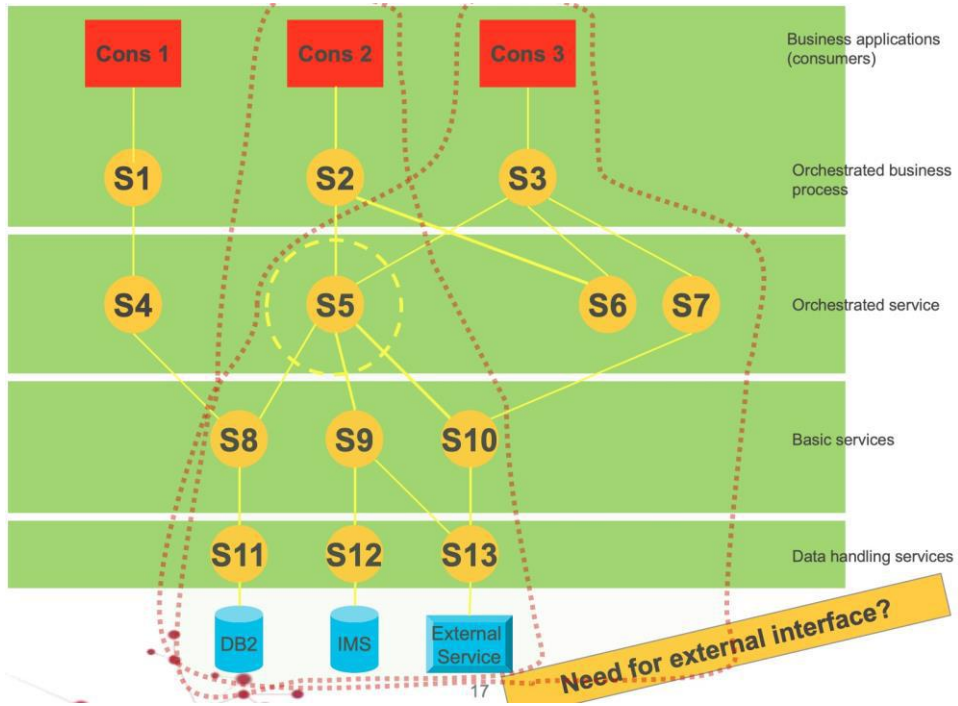
What must be tested?



Just the ones directly involved

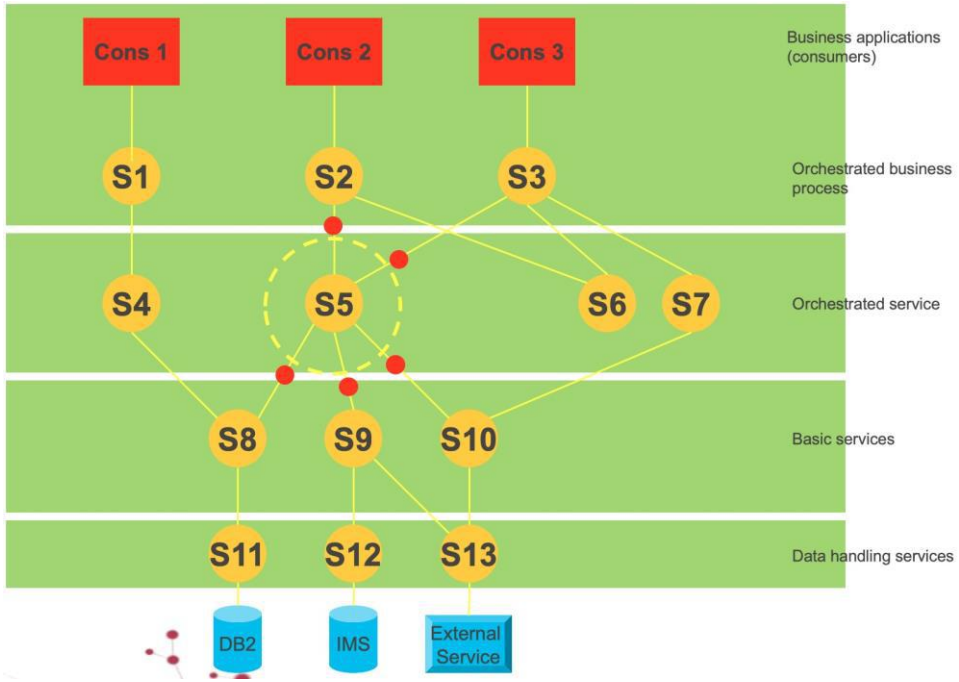


Pro’s and cons



Pro: Security**,** certainty

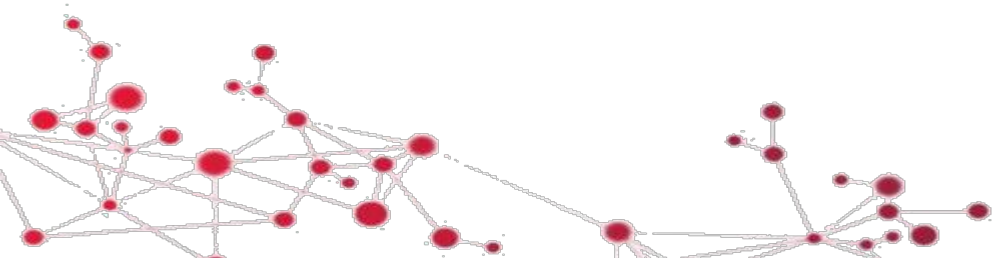
Con: Difficult to manage, expensive



Pro: Less effort, relatively easy to manage

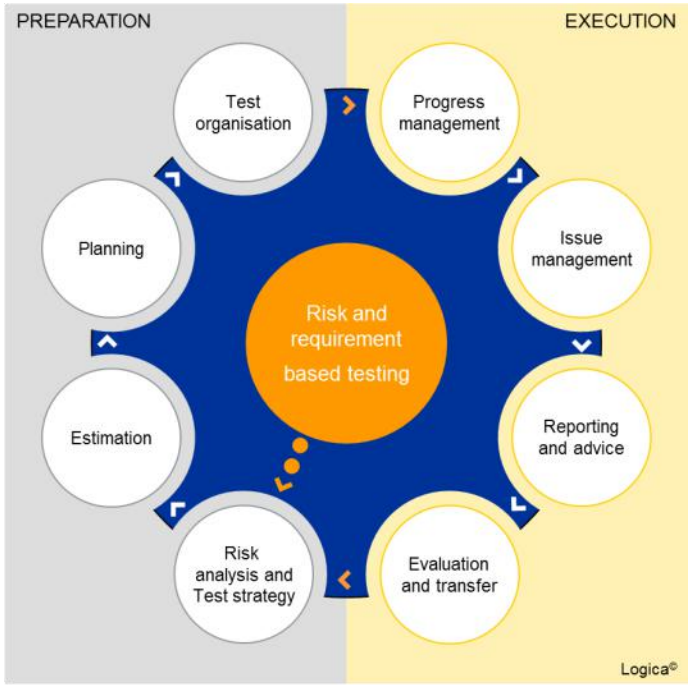
Con: Perhaps less insight in risks

Ideally, in "real SOA" there should be no need for end-to-end testing!



End-2-end testing: where do we stop?

- Chain testing is expensive and complex
- Stakeholders define the product risks
- Only high risk systems justify the effort for chain testing
- In other cases stick to interface based testing!



Product Risk Assessment as starting point

Test environments and test data management

How to deal with connections to non-existent services or services that are under change

Mock services (Stubs and Drivers)

- Specific pieces of software
- Simulate behaviour of services not (yet) available
- Allow for isolated testing of services
- Divide the chain into manageable parts
- Are adaptable to the wish of the tester
- Provide good control of data
- Good and fault situations are predictable

Mocks are interchangeable with virtualization tools

Stubs or Service virtualisation?

Mocks

- Initially cheap
- “Tailor made”
- Rapidly deployable
- Risk of expensive maintenance

But:

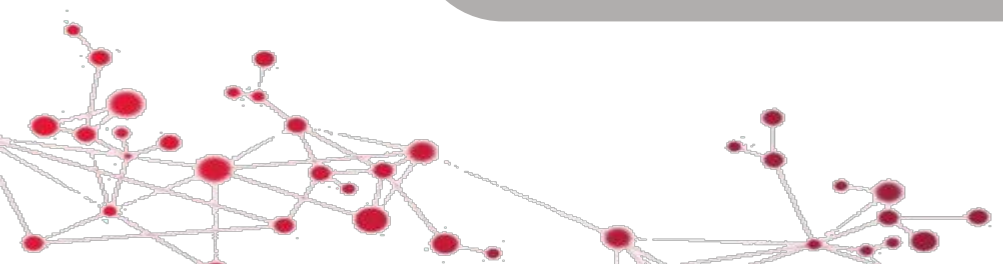
- Made by developers
- Additional, non-standard maintenance
- Additional communication tester <> developer
- Inflexible
- Poorly reusable

Virtual Services

- Every service can virtually be available
- Flexible and adjustable test data
- Maintained by testers
- High degree of reuse

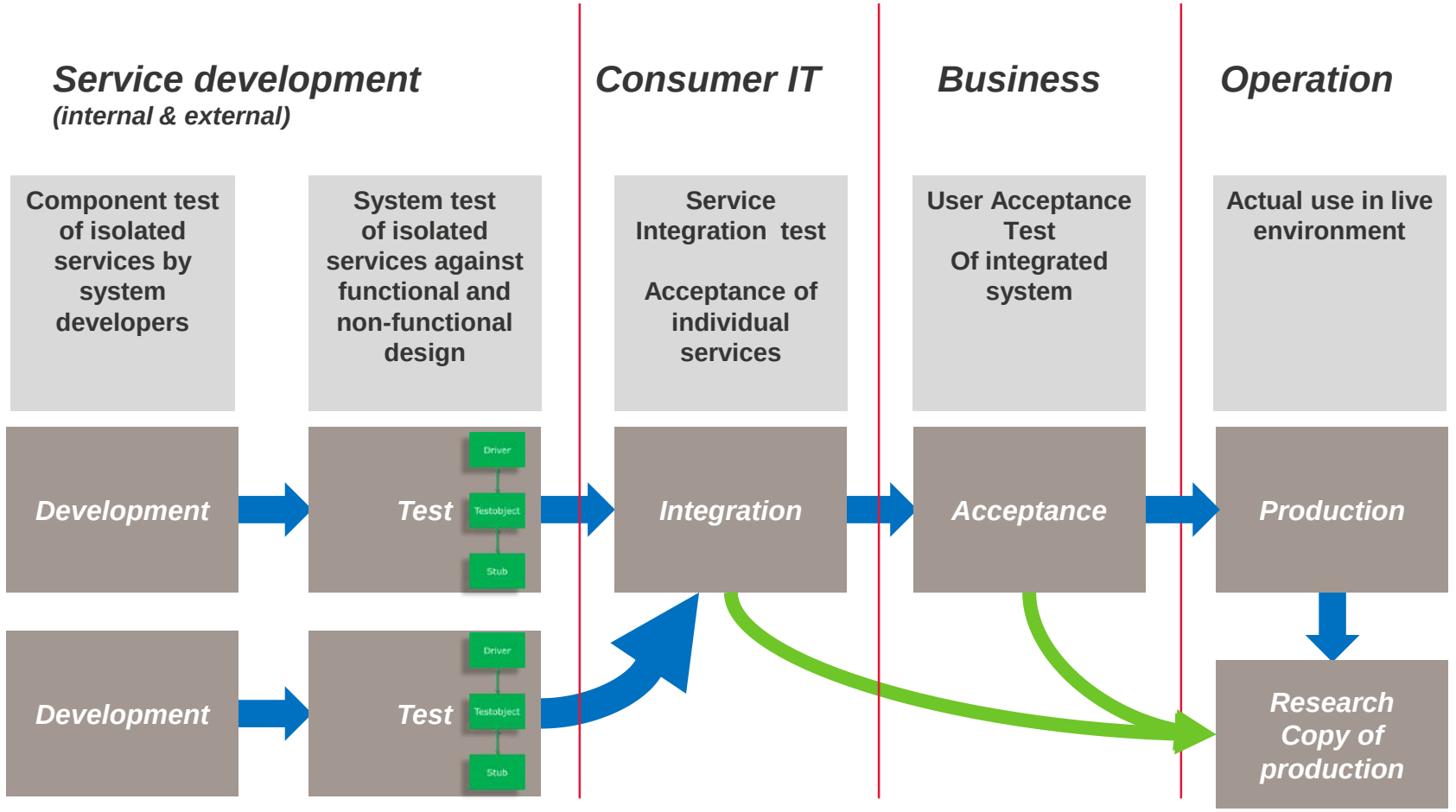
But:

- Relatively expensive (additional tooling required)
- Additional (functional) management
- Knowledge and training needed



CGI

Alternative: Several test environments



Role of the Test manager

- Is responsible for all test activities
- Compiles a test strategy based on product risks analysis
- Defines requirements to test environments
- Formulates demands on service delivery

Test manager is often also "Integration Manager"

First level Acceptance

- Does the service meet the requirements as stated in the repository?
- Is backwards compatibility guaranteed?
- Can the service safely replace the one that's already in the registry?

Fist level acceptance by the service owner

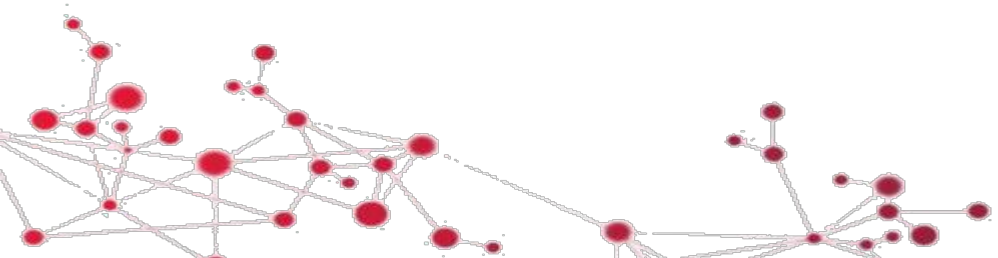
Today’s Agenda

01 Introducing SOA

02 Test specific items

03 SOA governance aspects

04 The real life experience



SOA Governance

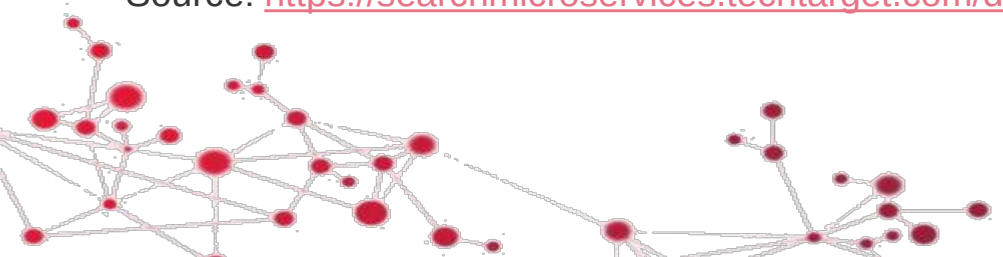
SOA governance is a concept used for activities related to exercising control over services in a service-oriented architecture (SOA).

IBM and others, state that **SOA governance** is an extension (subset) of IT governance which itself is an extension of corporate governance.

Source: https://en.wikipedia.org/wiki/SOA_governance

SOA governance refers to the processes used to oversee and control the adoption and implementation of service-oriented architecture (SOA) in accordance with recognized practices, principles and government regulations. **SOA governance** provides optimum service quality, consistency, predictability and performance, ensures that personnel follow prescribed policies and corrects system problems or policy infractions as they occur.

Source: https://searchmicroservices.techtarget.com/definition/SOA-governance

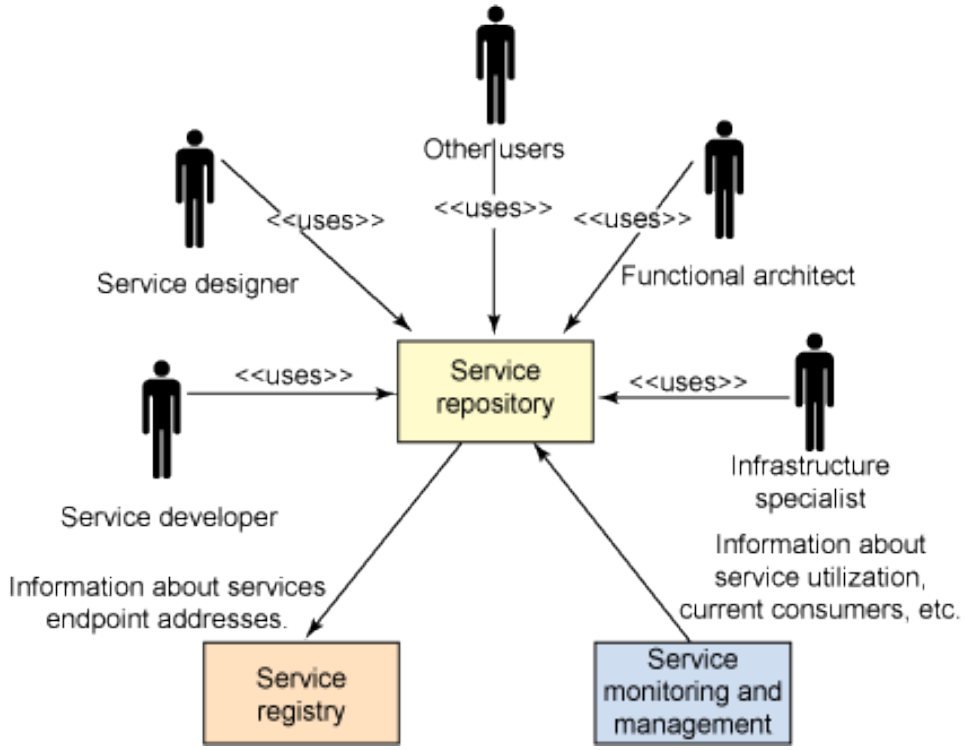


Service repositories and service registries

Two important aspects of SOA Governance

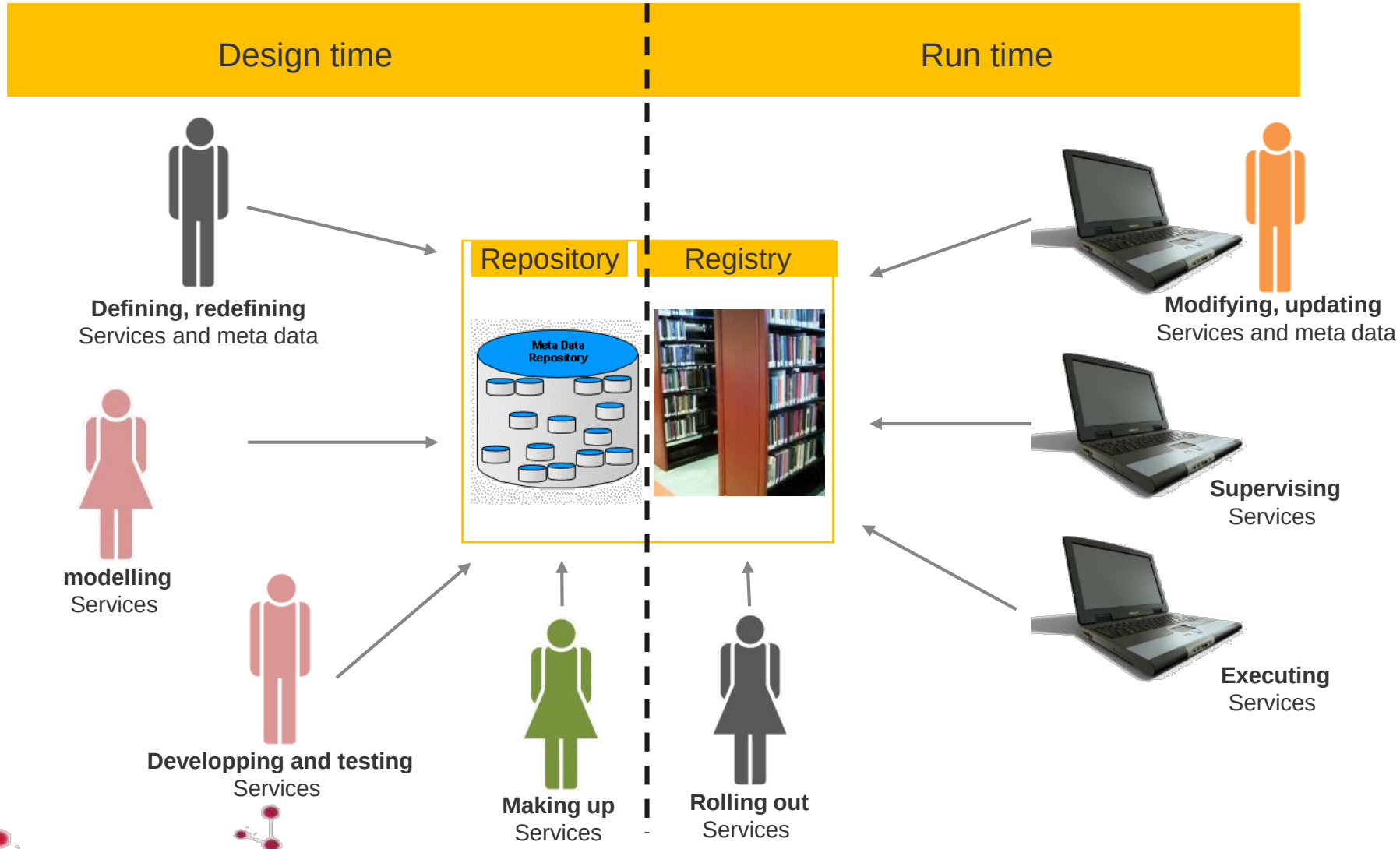
A ***service repository*** is the foundation of enterprise SOA governance, supporting centralized management of all of services-related information, including design, implementation, and usage artefacts.

A ***service registry*** is the foundation of service location virtualization, allowing for centralized control over location and invocation policies for all of the enterprise services.



Source: Boris Lublinsky - Explore the role of service repositories and registries in Service-Oriented Architecture (SOA) IBM-2007

Use of registry and repository

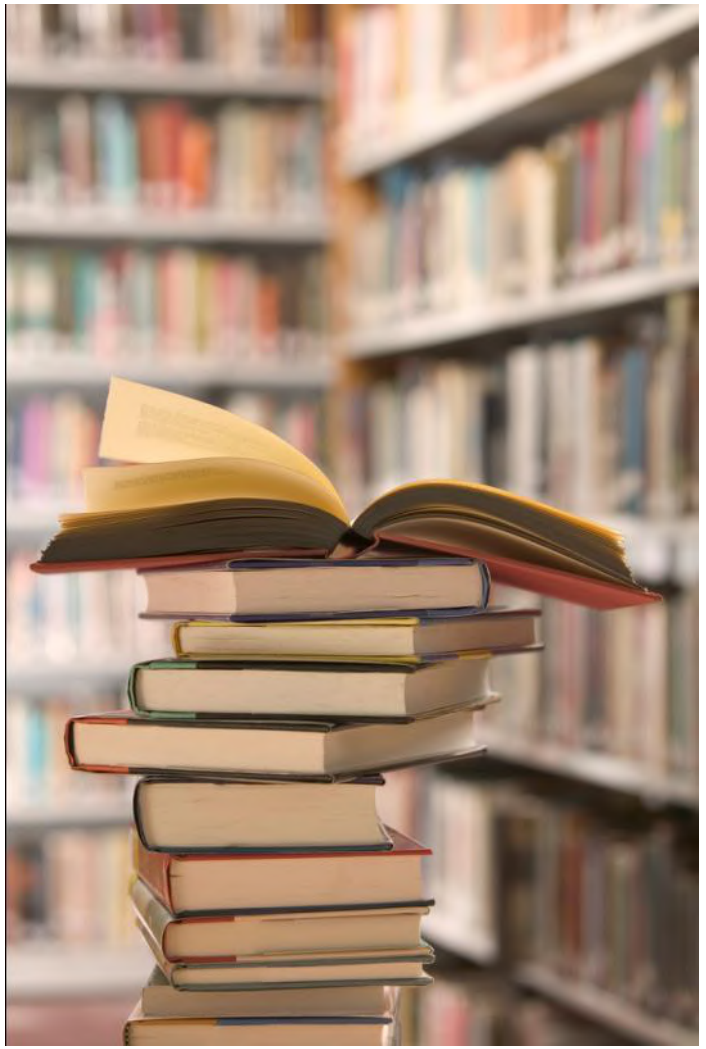


Stakeholders and the service repository

The details about your services and the services you use:

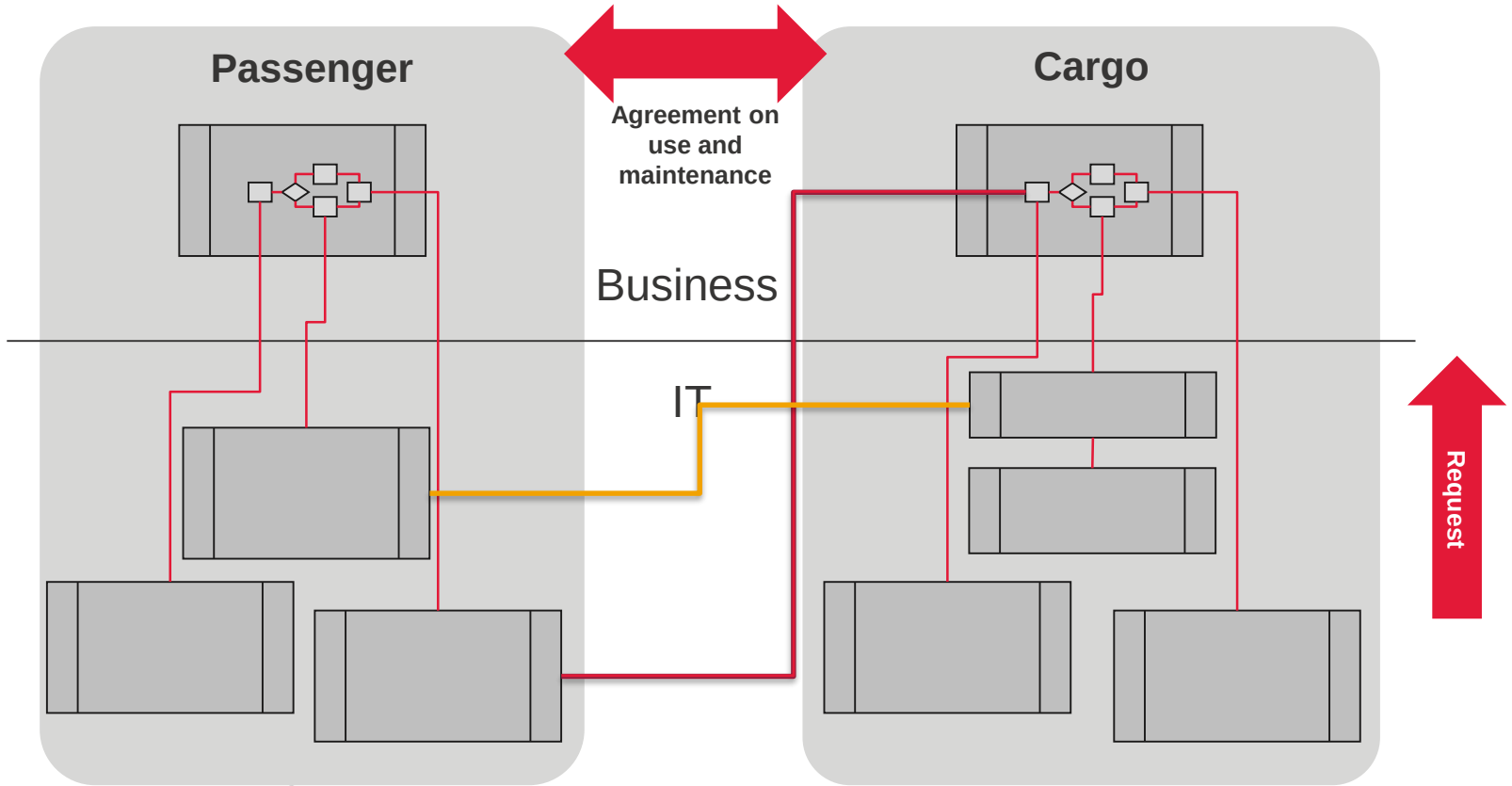
- Functionality
- Availability
- Capacity
- Use
- Service owners
- Service Sepcifications

SLA between suppliers and consumers



Business model

who will pay for the changes and test them if one consumer needs them and others don't.



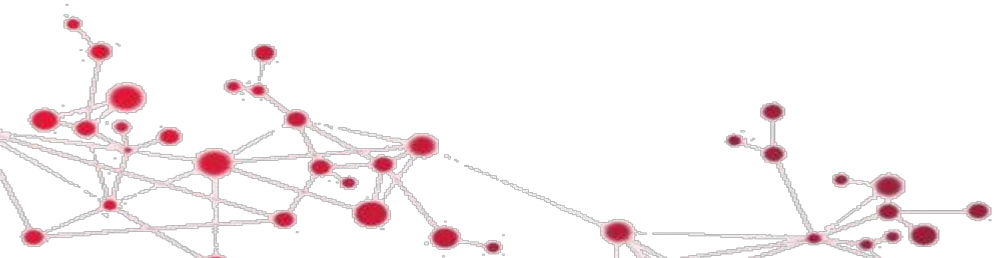
Today’s Agenda

01 Introducing SOA

02 Test specific items

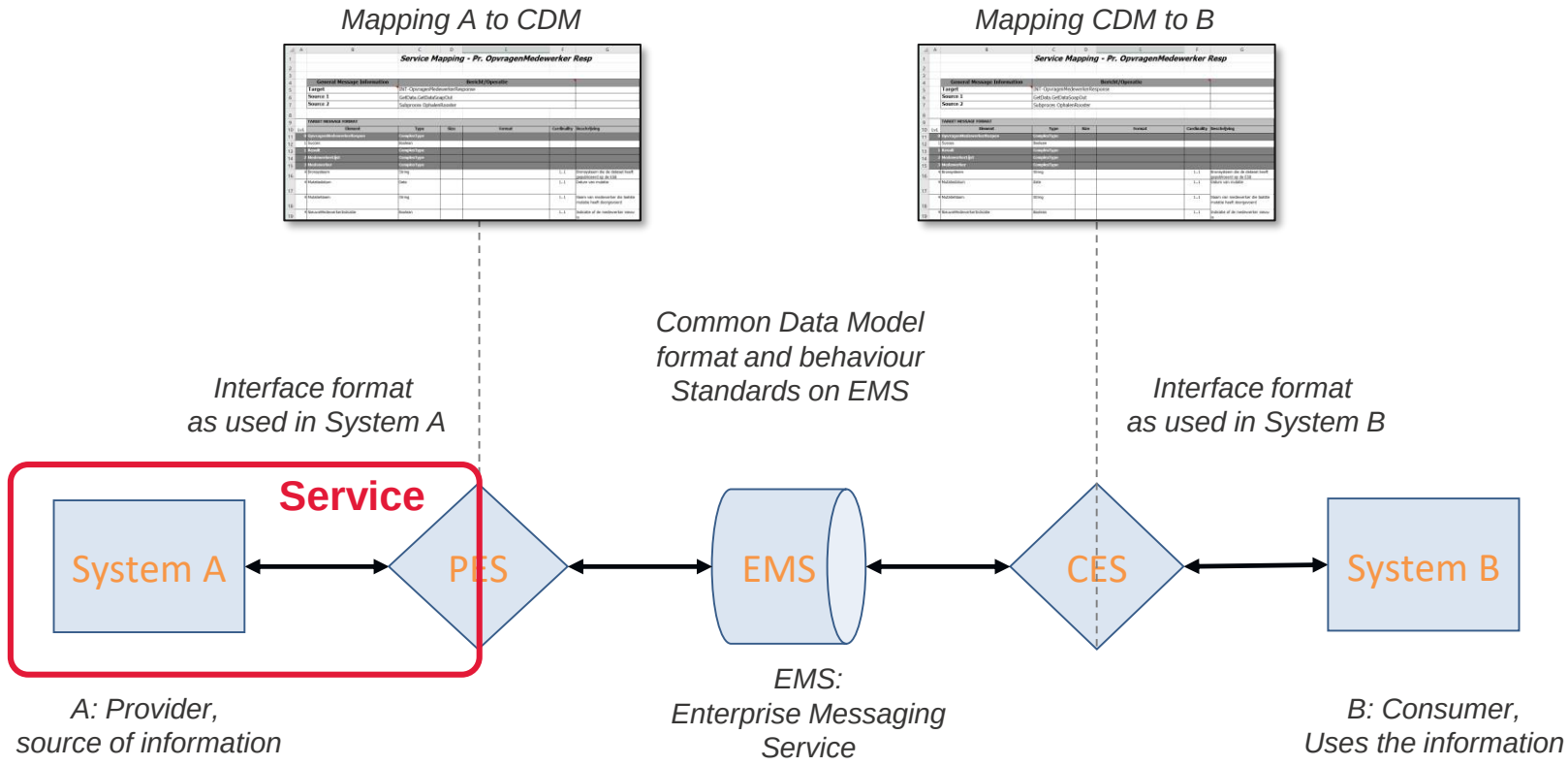
03 SOA governance aspects

04 The real life experience



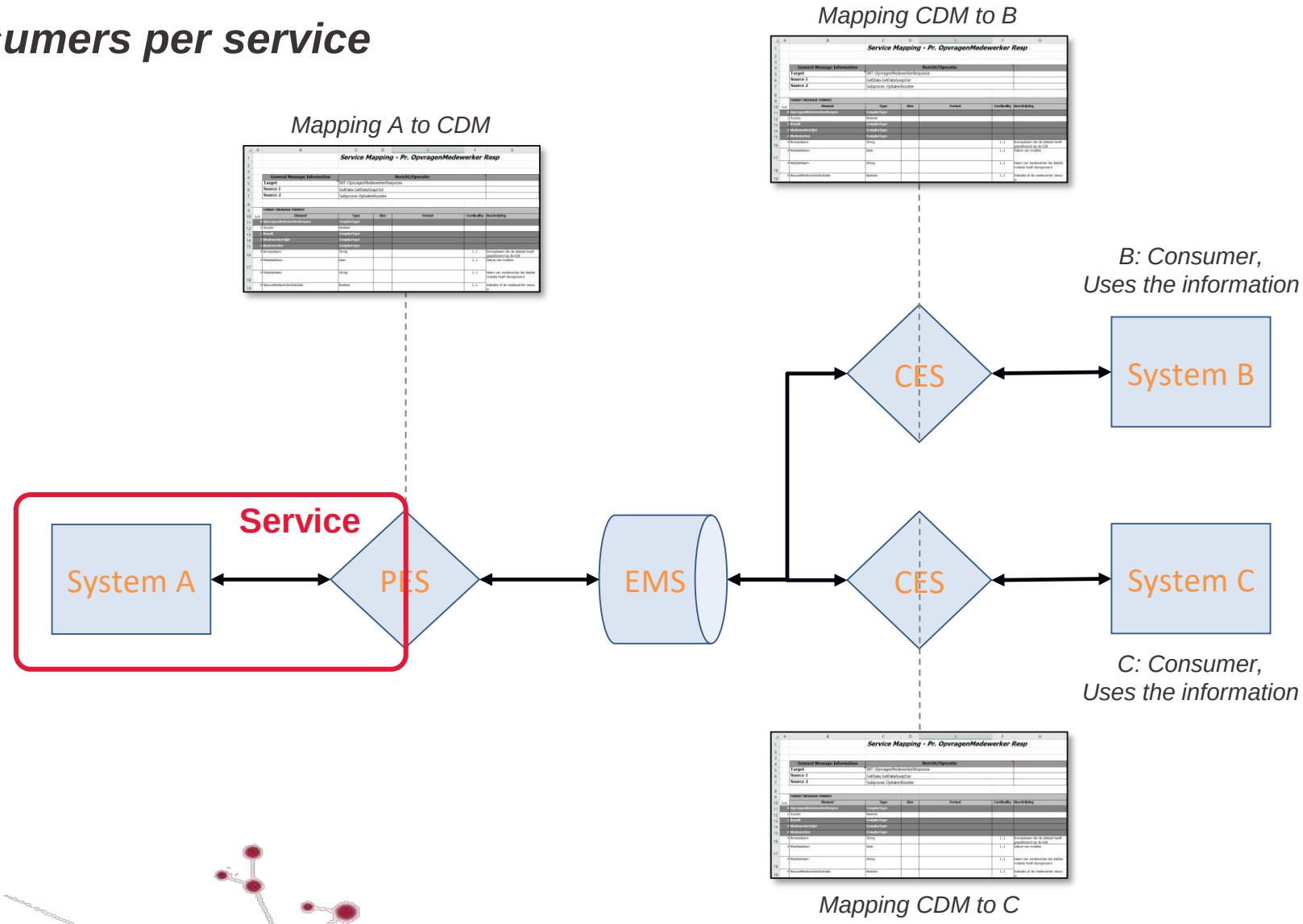
Testing of services in a live environment

The generic presentation of components in an Enterprise Service Bus (ESB)

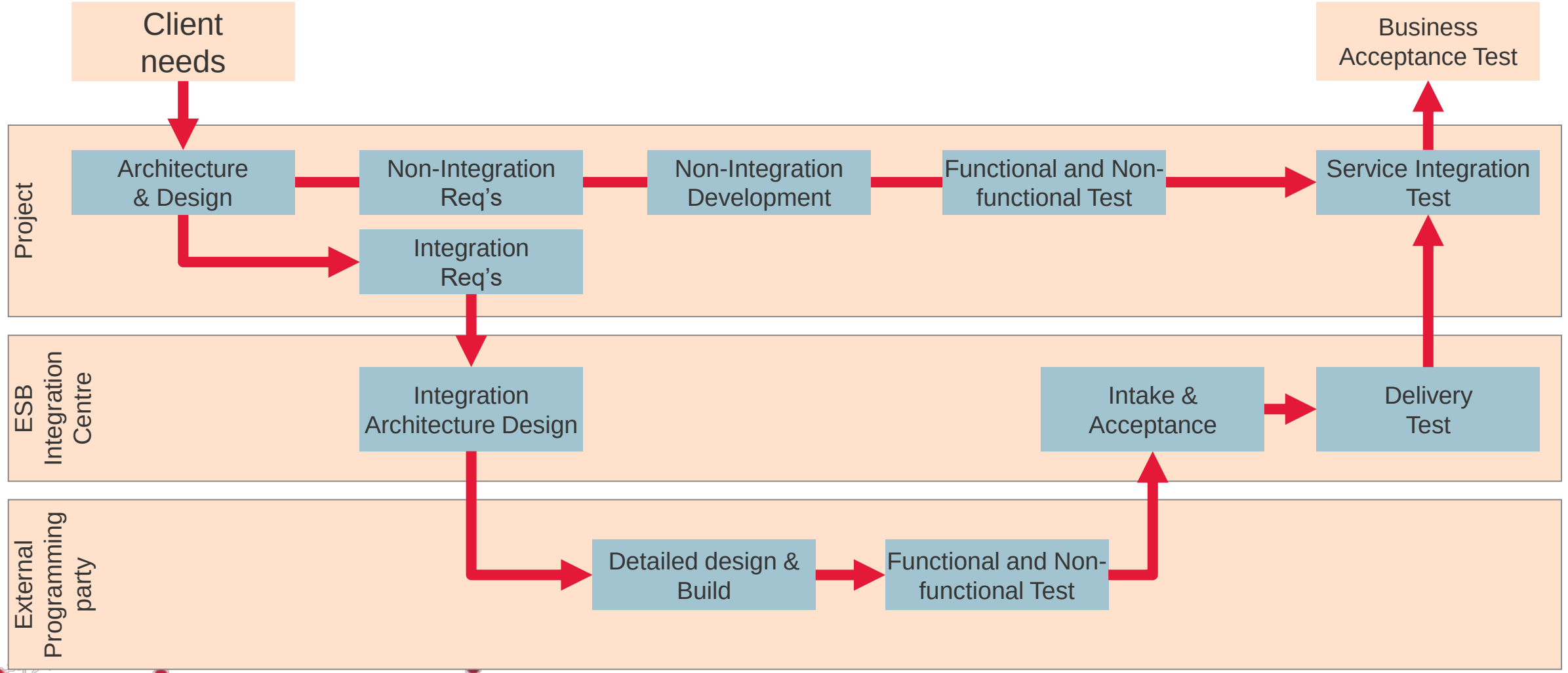


Testing of services in a live environment

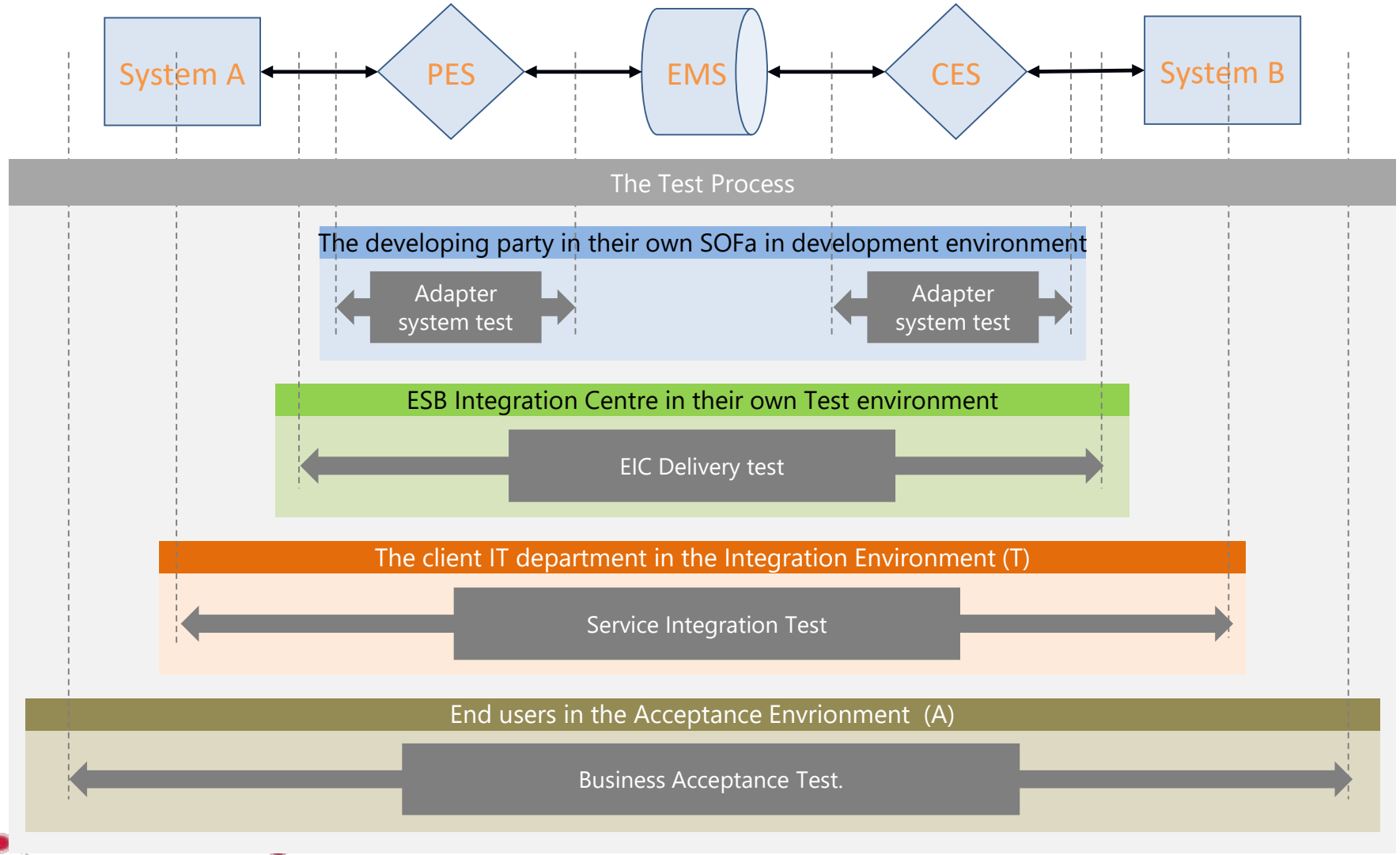
Multiple consumers per service



System development related to SOA



Testing of services in a live environment

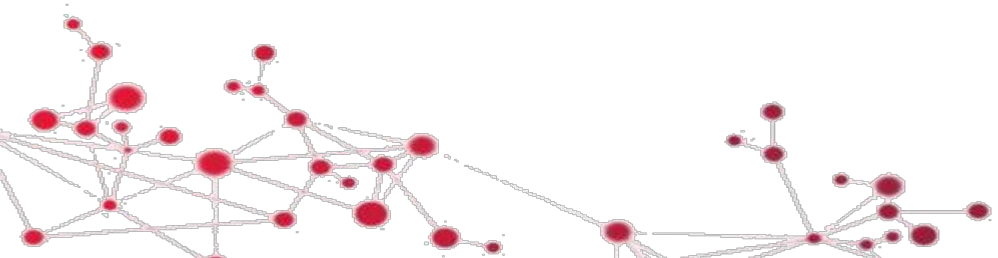


Generic Test Agreements

Overall test strategy and agreements

- Clear agreements on who test what and when
- Overall acceptance criteria per test phase
- Handover criteria between test phases

Like any test strategy: It prevents "I did not mean or expect that" discussions



Summary: testing in SOA environments

Accepting parties

- The users accept the application
- The service-owner accepts individual SOA services on functionality and capacity
- Operations accepts SOA services op load, stress and behaviour

Integration testing

- Preferably in isolation
- End-to-end if stakeholders consider it necessary and product risks indicate to

Regression testing

- All owners of services that communicate directly with the changed one
- Owners of the changed services
- Operations

The test managers role

- Supervising the test process
- Advising the stakeholders
- Integration of services

chris.schotanus@cgi.com
chris.schotanus@zoho.eu