Einsatzszenarien von SOA:
Drei Praxisbeispiele

Eine Präsentation für SI-SE

Philipp H. Oser und Michael Schröder

16.03.2007

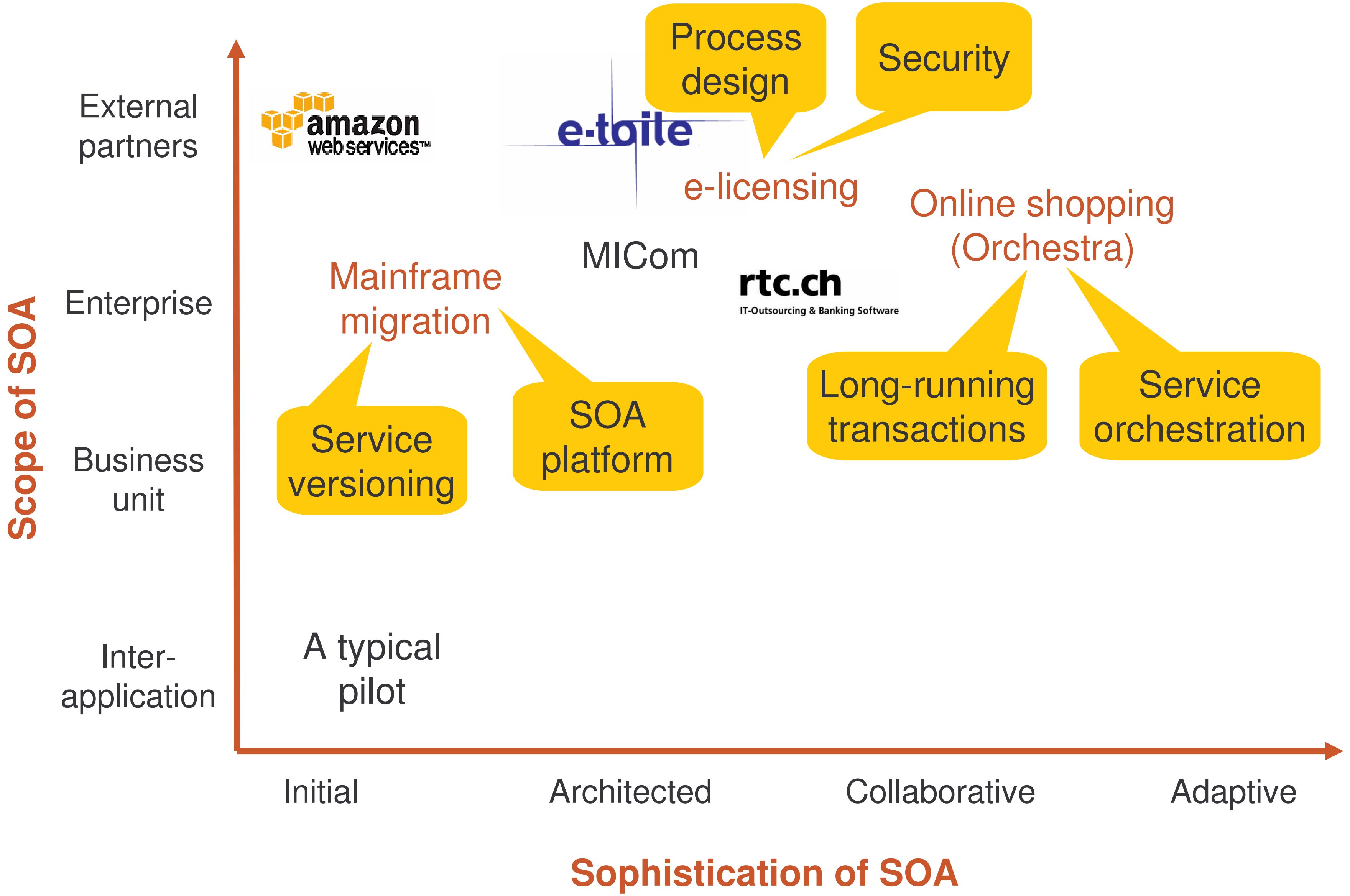
ELCA *We make it work.*

One dimension of SOA: “Sophistication of SOA“

“Sophistication of SOA“ →

	Initial	Architected	Collaborative	Adaptative
Characteristics	▪Few services ▪Simple usage scenarios ▪Single transport ▪Basic profile	▪Many services ▪Single transport ▪Ad-hoc transactions ▪Extended usage scenarios	▪Many transports ▪Many profiles ▪Service composition	▪Service orchestration / choreography
Application design	▪Apps use assets wrapped as services	▪Apps increasingly exchange via service reqs	▪Most business functions available as services	▪Apps are continuously recomposed
Elements	▪SOAP, WSDL ▪Adapters	▪Metadata registry	▪Transactions ▪Routing ▪Management ▪Security	▪Orchestrator ▪BPM notation and tools ▪Events

SOA pattern landscape



Three case studies: Overview

[Image source: IBM/WWW]

Case study 1:
Mainframe migration

Case study 2:
e-licensing (BAKOM)

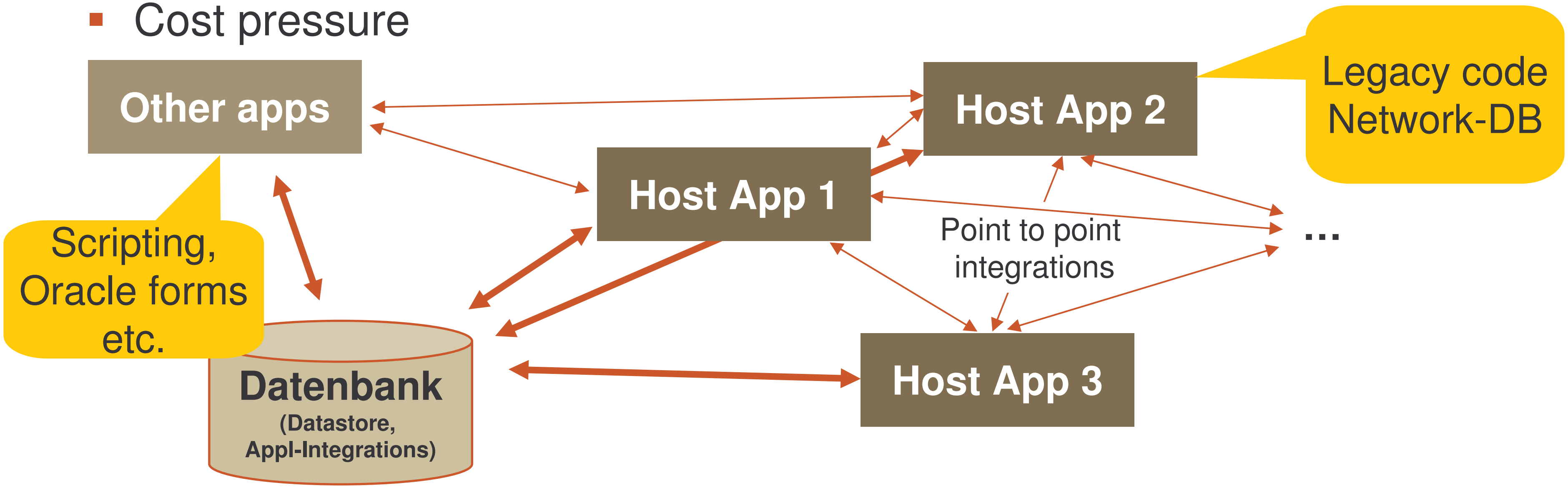
Case study 3:
Online-shopping (Orchestra)

Case-study: Migration from Mainframe

Domain: Financial management

Initial situation:

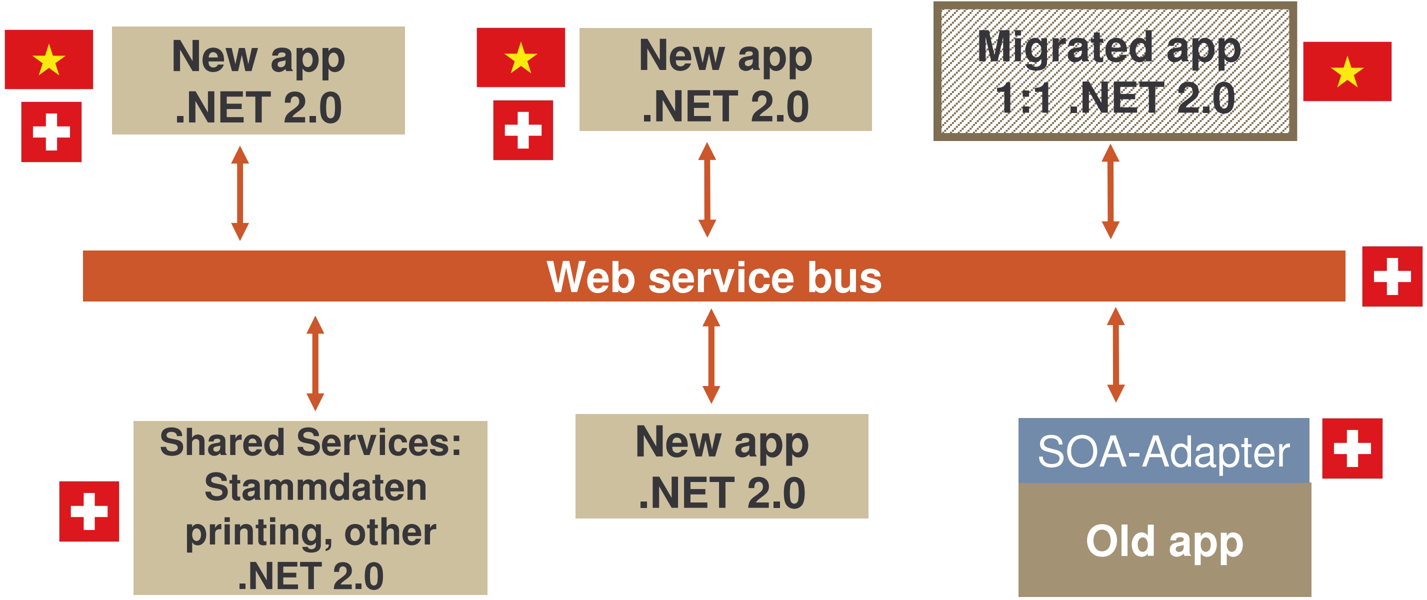
- Application portfolio mainly based on host technology, heterogeneous technologies
- Pressure to move away from the host due to ending vendor support
- Ad-hoc integration via database or point to point solutions
- Desire to keep some of the huge existing application know-how
- Cost pressure



Case-study: Migration from Mainframe

Target architecture:

- Better modularisation of application portfolio
- More structured, state-of-the-art integration with less coupling
- Use existing application assets
  - Wrapping
  - 1:1 application migration
- Pragmatic SOA platform (SOA for integration)



Web service versioning

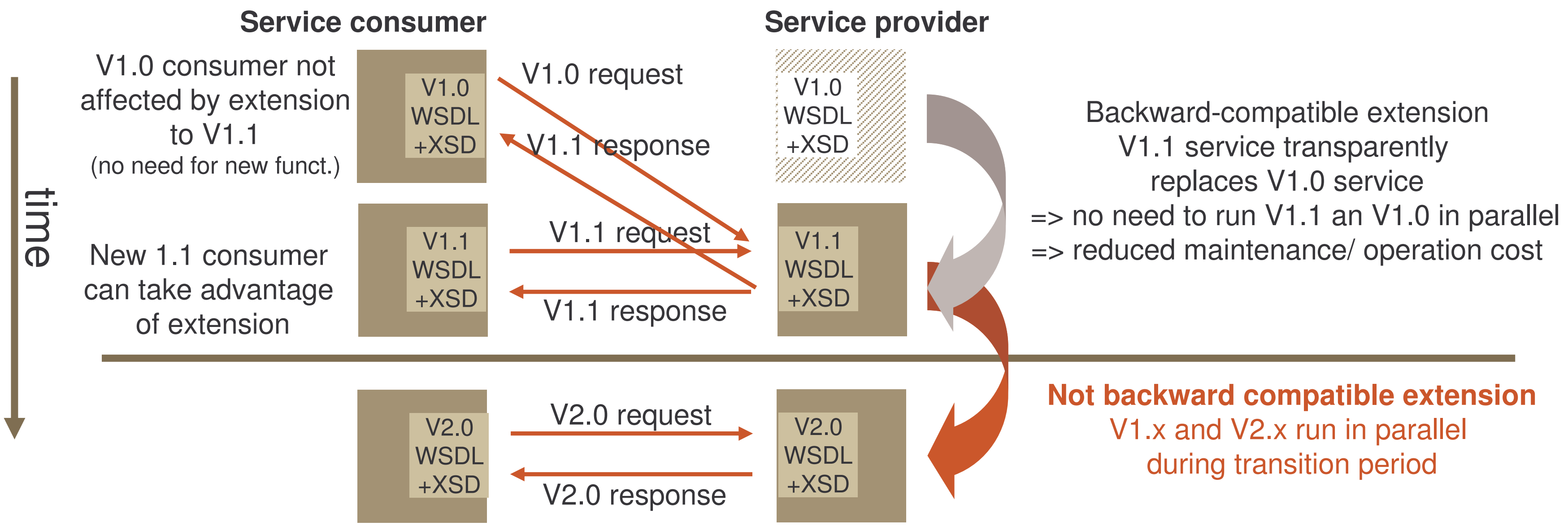
For flexibility of service evolution

- Run multiple versions of a same service in parallel
- Do not require big-bang upgrades

Running many service versions in parallel can be costly

- Operation cost and complexity
- Bugfixes may need to be applied to all productive versions

Solution: Some **version backward-compatibility** can help!



Service consumer
Service provider
time
V1.0 consumer not affected by extension to V1.1
(no need for new funct.)
New 1.1 consumer can take advantage of extension
V1.0 WSDL +XSD
V1.1 WSDL +XSD
V2.0 WSDL +XSD
V1.0 request
V1.1 response
V1.1 request
V1.1 response
V2.0 request
V2.0 response
Backward-compatible extension
V1.1 service transparently replaces V1.0 service
=> no need to run V1.1 an V1.0 in parallel
=> reduced maintenance/ operation cost
Not backward compatible extension
V1.x and V2.x run in parallel during transition period

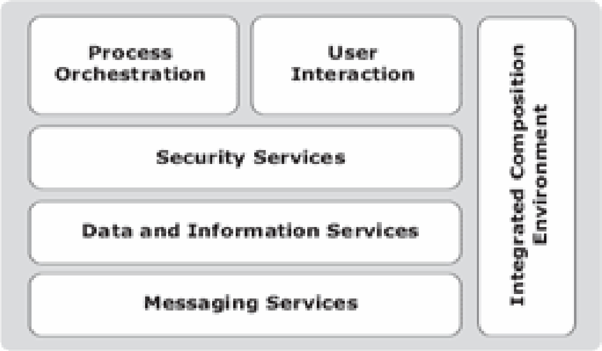
Platforms for SOA: General

Today's primary development environments (J2EE & .NET) have far-going support for web services

- Implementation of individual service is easily doable

Question: Add an **Enterprise Service Bus (ESB)** to your platform?

- Another question: Level of needed non-functional support?
- Main ESB features: Declarative policies (external to application) for
  - Security
  - Message Routing
  - Message translation (e.g. for versioning, interop)
  - Orchestration
  - Management
    - Service & Metadata registry
    - SLA-checks
    - Supervision



A typical ESB stack

Platforms for SOA: ESB

Shall we use an ESB stack?

Advantages (+)	Disadvantages (–)
▪ Systematic resolution of non-functional and of management issues ▪ Policies are outside of application ▪ Changes are easier possible ▪ Integration with different services easier (less non-functional incompatibility)	▪ Vendor/product dependency ▪ Complexity

Alternatives:

- Open-source ESB stacks (e.g., Mule, ServiceMix, Celtix)

Gartner: „ESBs will catch on broadly“

Three case studies: Overview

[Image source: unknown]

Case study 1:
Mainframe migration

Case study 2:
e-licensing (BAKOM)

Case study 3:
Online-shopping (Orchestra)

Case-study: e-licensing

Domain: E-Government (BAKOM)

Initial situation:

- Need for a new G2B/G2C application for selling of radio licenses over Internet
- Desire to reuse existing business applications

eOFCOM - Virtueller Schalter

Goals:

- For clients:
  - Easier to get licenses, application is available around the clock, faster processes
- For BAKOM:
  - More efficient selling processes, partly or fully automated
  - Support the Swiss e-Government Strategy

Case-study: e-licensing

Target architecture:

- The existing business applications are refactored to *business services* and integrated in a generic platform of BIT (Bundesamt für Informatik und Telekommunikation). The integration is via a set of generic services:
  - Authentication
  - Identity-Management
  - e-Payment
  - Service to access the SAP backend
- The first phase is a pilot – experiences are used for further phases
- Implications for application portfolio
  - (E-Government-)Applications will be composed of generic components and business components.
  - Business functionality will be only developed once, data will be stored centrally.

Case-study: e-licensing Business Case

Size metrics

- 70‘000 orders per year (>300 daily)
  - 40‘000 licenses for radio, networks and media
  - 30‘000 attributions of addressing- and numbering elements (70% of which are already done via Internet)
  - 1‘000 notifications of equipment

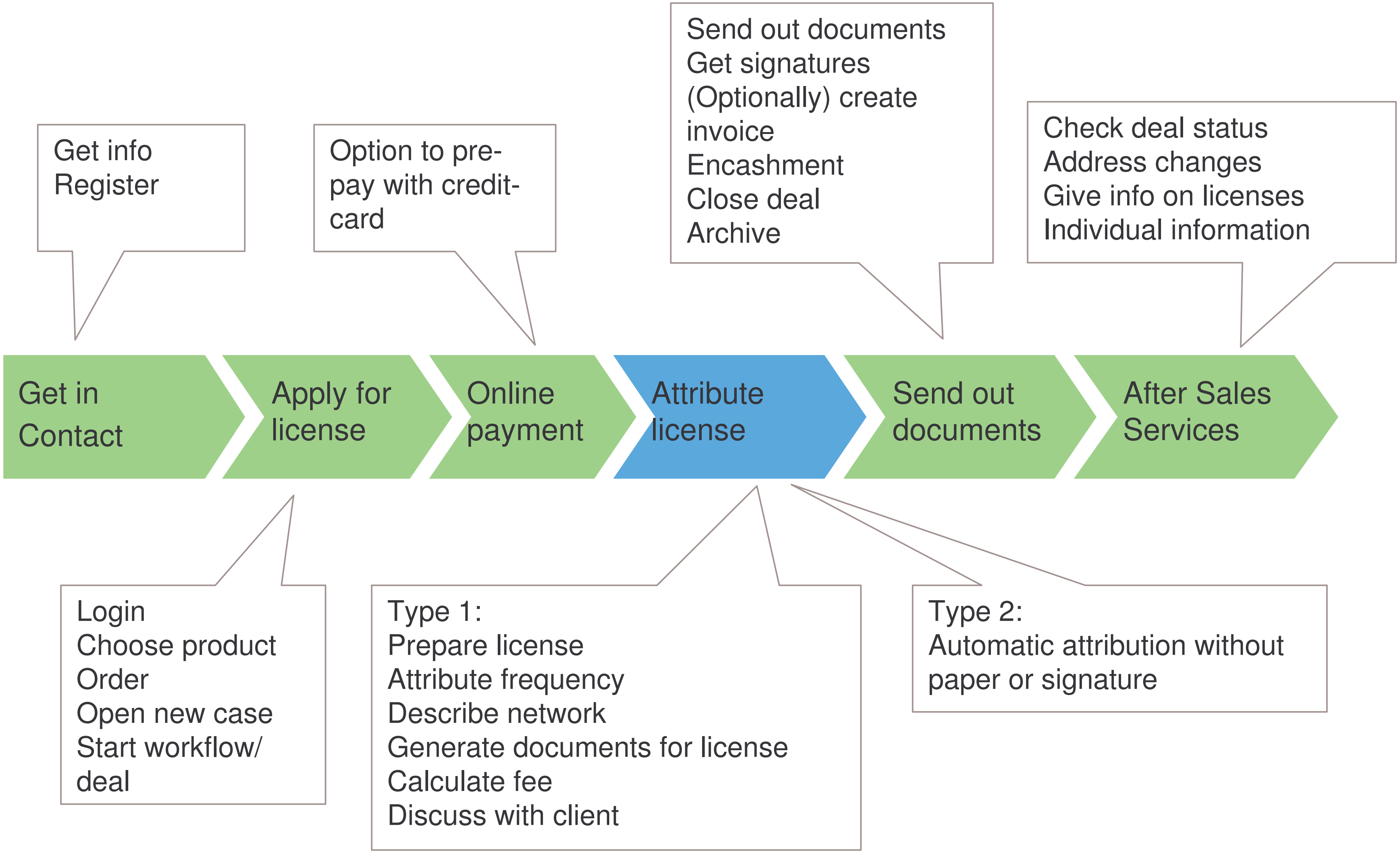
Business Case

- Significant speed up of licensing process
- ROI within one year

Expected IT cost reduction

- No short-term IT cost reduction
- Significant IT cost reduction for investments and operations through architecture scalability and reuse in the medium-term.

Case-study: e-licensing   Business Process



Get info
Register

Option to pre-pay with credit-card

Send out documents
Get signatures
(Optionally) create invoice
Encashment
Close deal
Archive

Check deal status
Address changes
Give info on licenses
Individual information

Get in Contact

Apply for license

Online payment

Attribute license

Send out documents

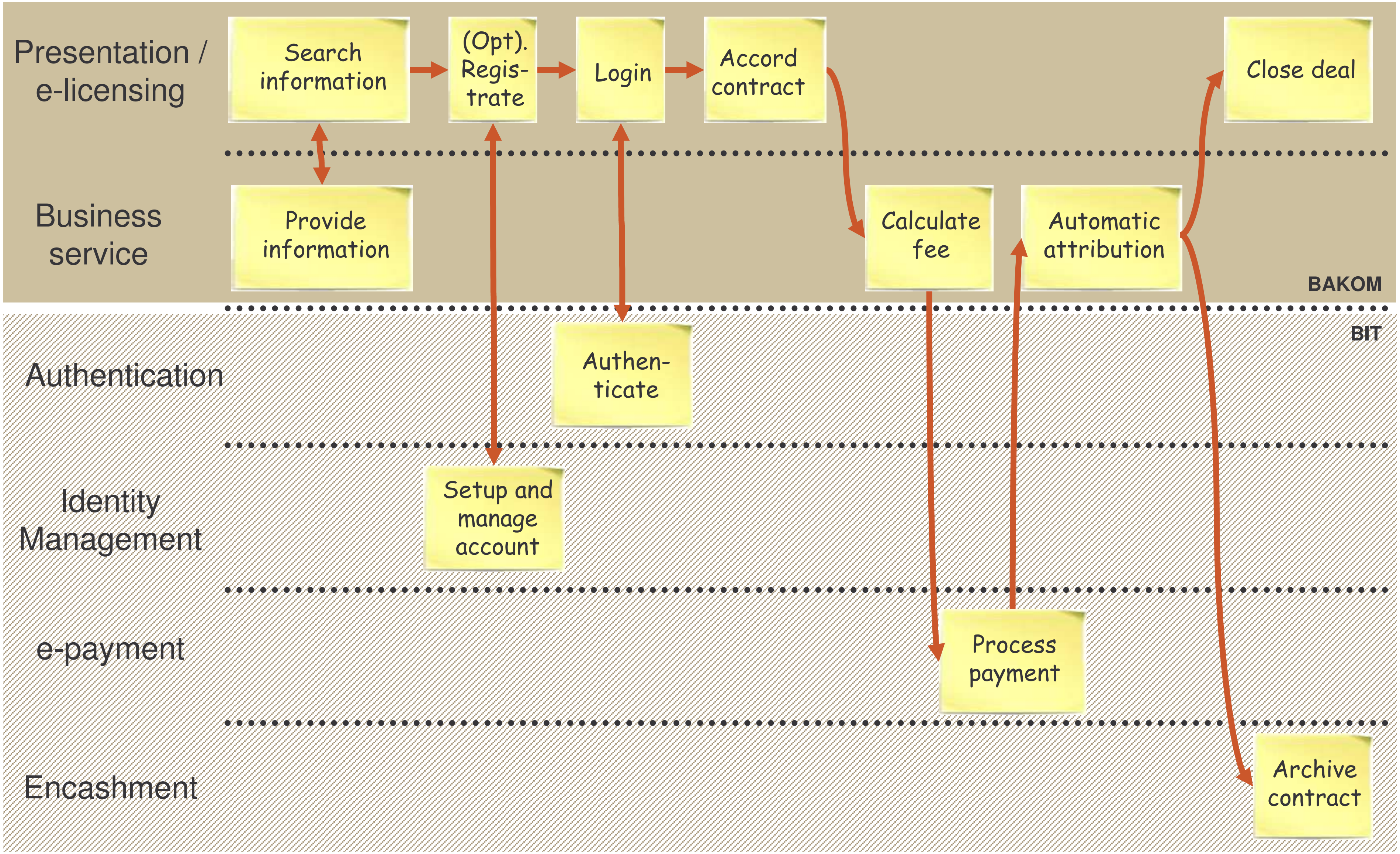
After Sales Services

Login
Choose product
Order
Open new case
Start workflow/ deal

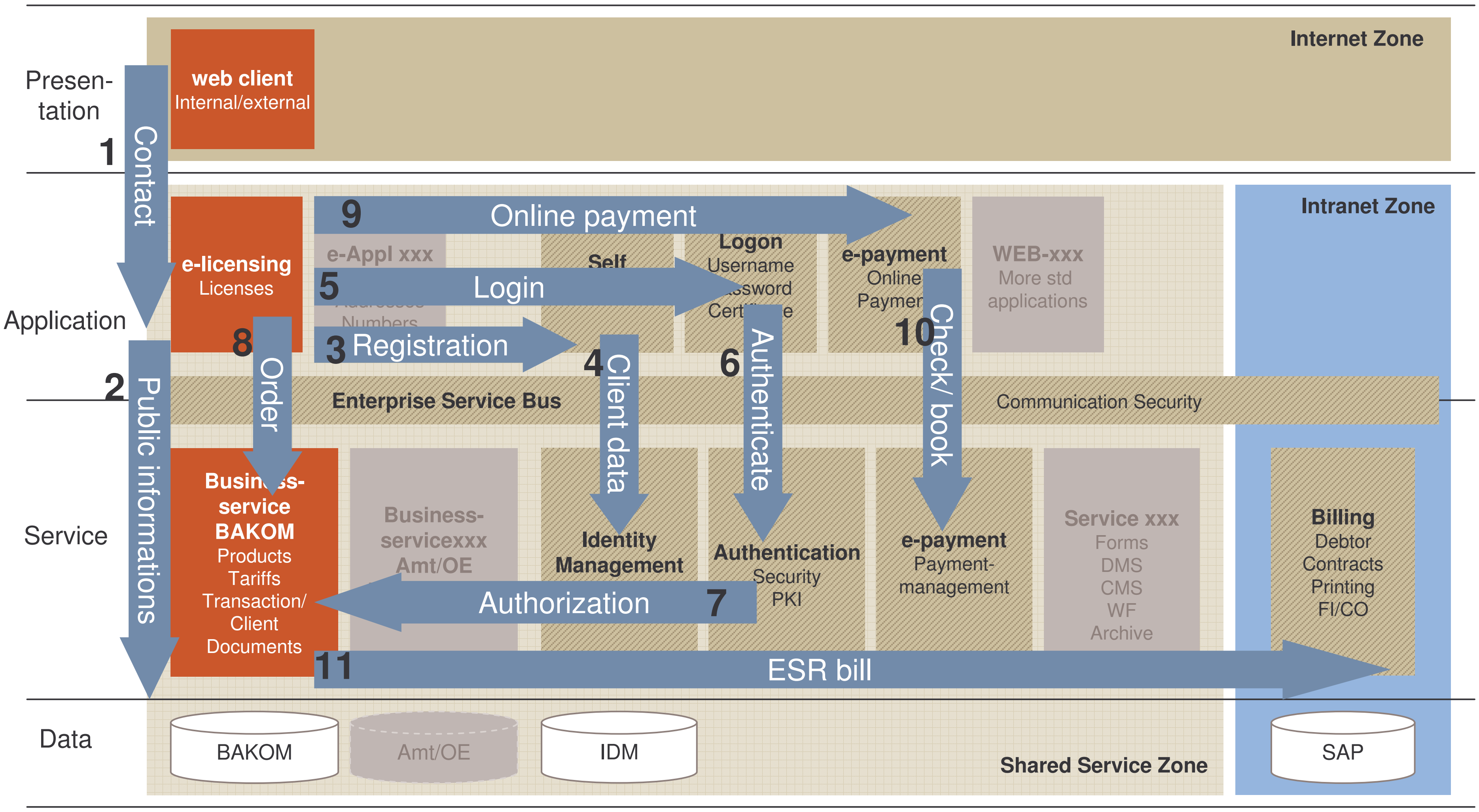
Type 1:
Prepare license
Attribute frequency
Describe network
Generate documents for license
Calculate fee
Discuss with client

Type 2:
Automatic attribution without paper or signature

Case-study: e-licensing Distribution of Process



Case-study: e-licensing Logical Architecture



Case-study: e-licensing   Challenges

Optimise internal learning curve

- Architect has role on level business, application and IT-infrastructure
- A Service is no longer only a technical component; it includes supporting business processes

Complexity management

- Many partners and stakeholders
- Tests and error detection
- Service release management

Separation of applications becomes more fluent

- SLA supervision becomes important
- New model for service billing

Securing web services: General

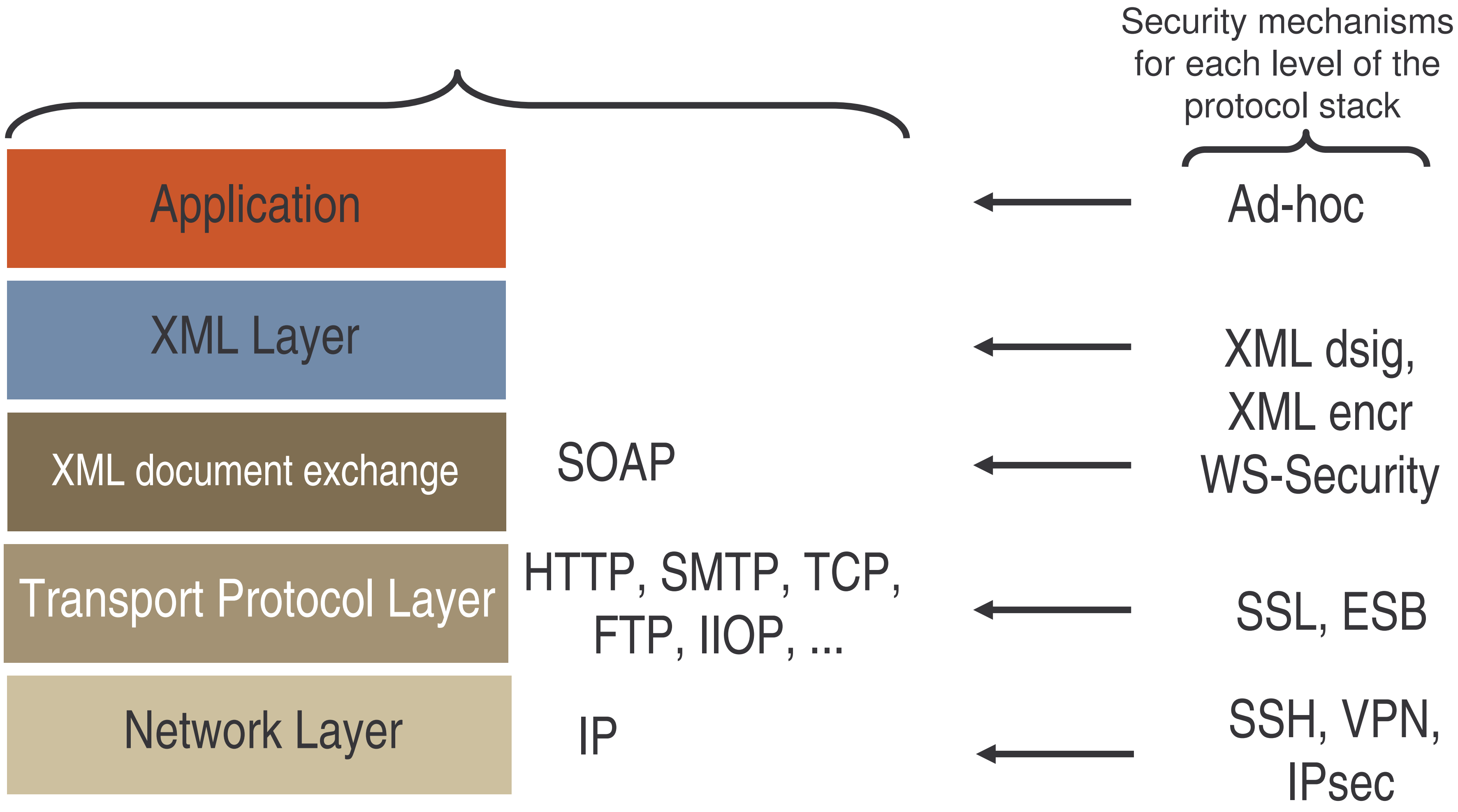
Critical when web services are used across administrative domains!

Security requirements for web services:

- Authentication
- Authorization
- Privacy
- Integrity
- Auditability
  - Proof of origin/ proof of receipt
  - Audit logs

Securing web services: Protocol stack

Webservice protocol stack with typical protocols



Security mechanisms for each level of the protocol stack

Application

Ad-hoc

XML Layer

XML document exchange

SOAP

XML dsig,
XML encr
WS-Security

Transport Protocol Layer

HTTP, SMTP, TCP, FTP, IIOP, ...

SSL, ESB

Network Layer

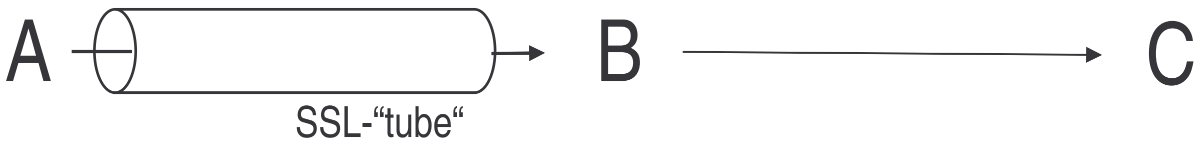
IP

SSH, VPN, IPsec

Securing web services: Pragmatically…

Transport Protocol Layer security, e.g., with HTTPS, SSL:

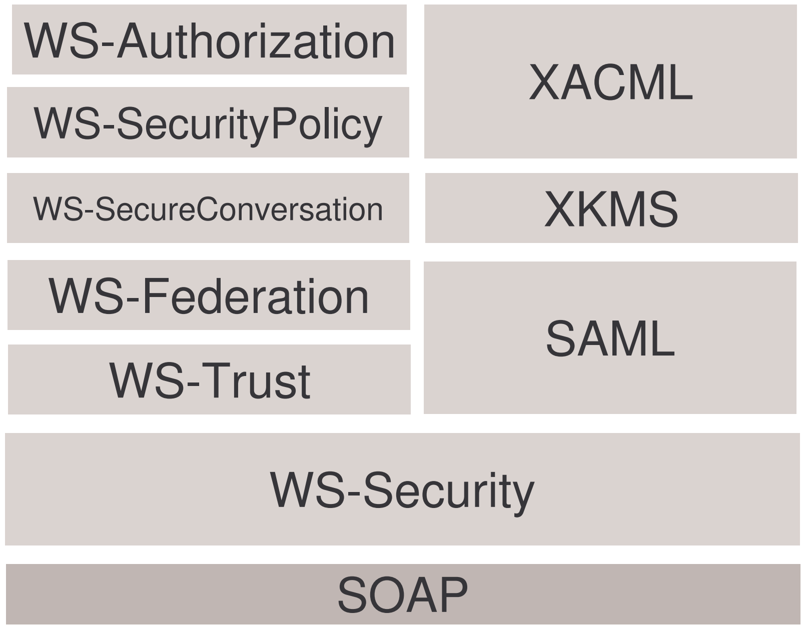
- Provides: authentication, confidentiality, and data integrity



Advantages (+)	Disadvantages (–)
HTTPS and HTTP basic authentication are widely available Simple and well known Widely implemented Ready for most transport protocols (e.g., HTTP, SMTP, IIOP)	Only for 1 hop (e.g., need full trust in first party that receives data) Limited security features: no proof of origin

Securing web services: …and via “Rolls-Royce”

WS-Security stack (one possible way to put it)



Established standards:
WS-Security, SAML

Advantages (+)	Disadvantages (–)
▪ Supports even the most demanding security scenarios: ▪ Delegation ▪ Multi-hop ▪ Selective security	▪ Complex ▪ Reduces compatibility as both partners need to have a compatible security stack ▪ Not yet completely specified and implemented

Three case studies: Overview

[Image source: OGM]

Case study 1:
Mainframe migration

Case study 2:
e-licensing (BAKOM)

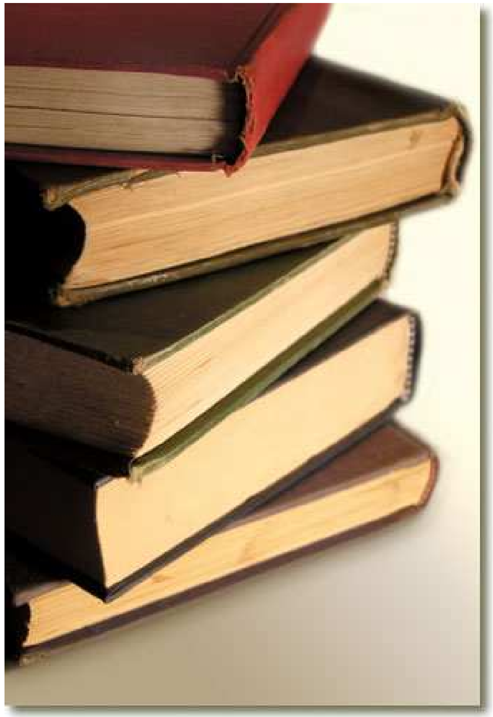
Case study 3:
Internet selling (Orchestra)

Case-study: Internet selling (Overview)

Domain: Platform to sell goods over Internet (partly confidential)

Initial situation:

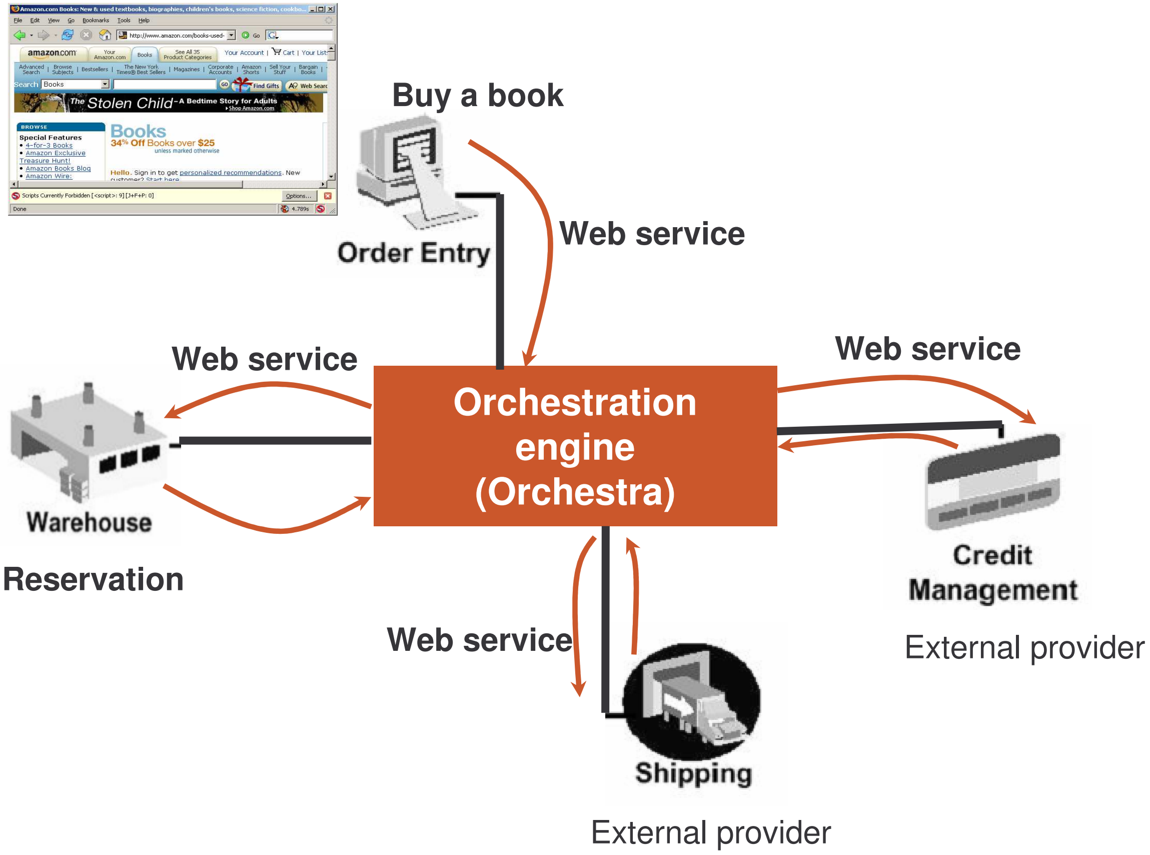
- Selling-platform is split in separate components
- Need for an integration platform that easily integrates with existing components in the environment of the customer
- Need for long-running and distributed transactions
  - Transaction semantic crucial, many reasons for rollbacks
  - Some components (payment, shipping) may run in other administrative domain (external partners)
  - Some of the external systems are not prepared to work with a global transaction manager



Case-study: Internet selling

Target architecture:

- Web service technology for integration
- Orchestration engine „orchestra“ as integration backbone
  - Simple and proven technology for high reliability
  - „Knows“ selling-processes
  - Concept of *relaxed transactions and undo operations*
  - End-points need no 2PC awareness
- Asynchronous integration for better decoupling



Amazon.com Books: New & used textbooks, biographies, children's books, science fiction, cookbo...

http://www.amazon.com/books-used-

The Stolen Child - A Bedtime Story for Adults

Books 34% Off Books over $25 unless marked otherwise

Hello. Sign in to get personalized recommendations. New customer? Start here.

Scripts Currently Forbidden [<script>: 9] [J+F+P: 0]

Done 4.789s

Transactional Integrity: General

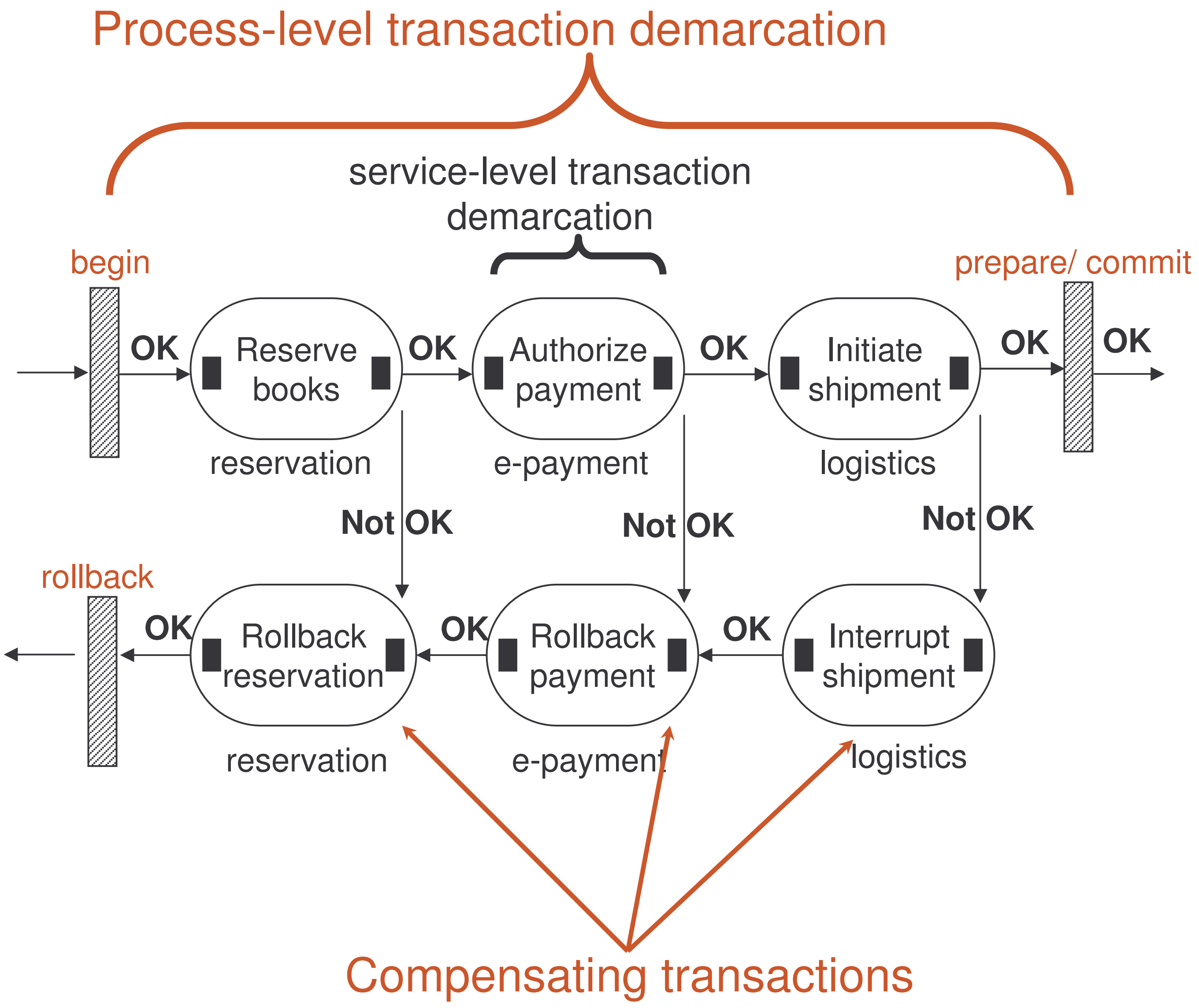
Strict ACID transactions (XA – 2PC) do not work with loosely-coupled and long running services:

- Locks required for a long time
  → hinders the concurrency in the system, DoS-attacks possible
- Loosely-coupled systems do typically not accept their transactions to be managed externally (e.g., payment system)
- Too much dependencies over provider boundaries (what if other system is down)?

Solution: *Relaxed transactions with undos*

- For each operation provide a *compensating* operation→ Is a second transaction (called rollback()) that cancels (undoes) the work done in the original transaction
- ACID for the top-level transaction is *relaxed*
  → during a certain *transitional* time, transactions are less isolated, not atomic, less consistent, not durable
- Orchestrator manages the transaction

Transactional Integrity: Relaxed transactions with undo



Compensating transactions

Three case studies: Overview

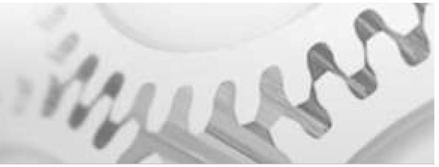
Case study 1:
Mainframe migration

Case study 2:
e-licensing (BAKOM)

Case study 3:
Internet selling (Orchestra)

Overview and summary

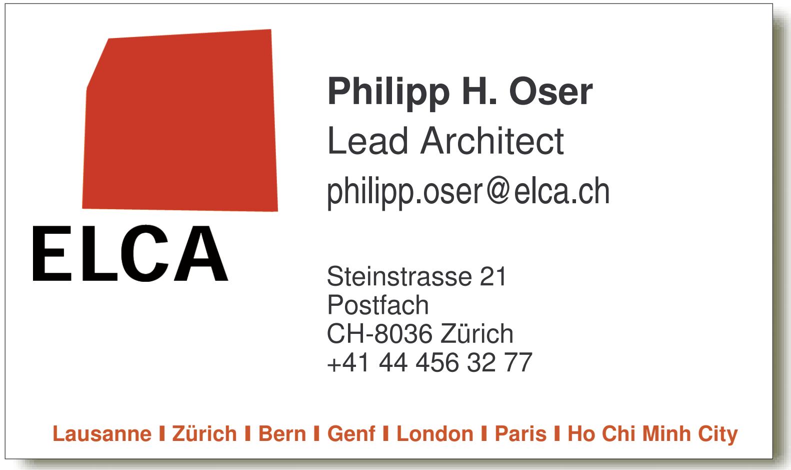
Recapitulation of the case studies



	Migration from Mainframe	e-licensing	Online-Shopping
Domain?	- Financial management	- E-Government	- Selling on the Internet
Why SOA?	- Modularisation of the architecture with state-of-the-art technology - End of service of mainframe / legacy	- Process execution across partners - Usage of generic services	- Need for integration architecture - Integration of yet unknown partner systems
Form of SOA?	- Pragmatic with features of .NET	- ESB for global policies - High security	- Long transactions - Orchestration engine
Challenges?	- Achieve performance of mainframe - Vietnam collaboration - Keep it pragmatic	- Organisational aspects	- High volumes & variance - Reliability even across administrative domain

Thank you for your attention

For further information please contact:

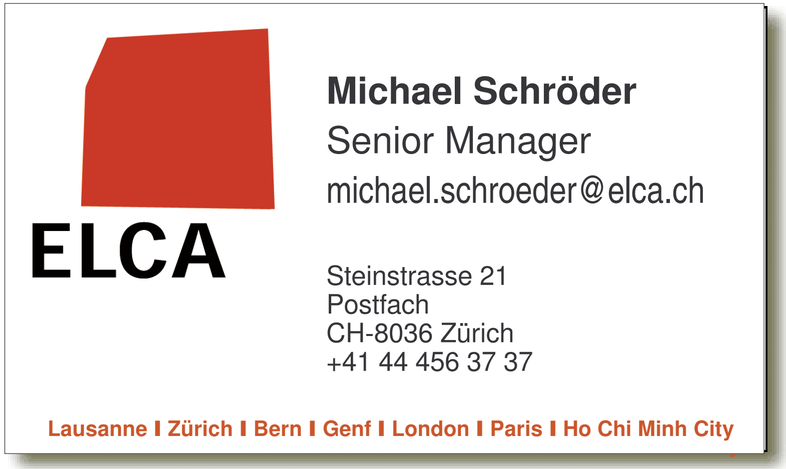


ELCA

Philipp H. Oser
Lead Architect
philipp.oser@elca.ch

Steinstrasse 21
Postfach
CH-8036 Zürich
+41 44 456 32 77

Lausanne I Zürich I Bern I Genf I London I Paris I Ho Chi Minh City



ELCA

Michael Schröder
Senior Manager
michael.schroeder@elca.ch

Steinstrasse 21
Postfach
CH-8036 Zürich
+41 44 456 37 37

Lausanne I Zürich I Bern I Genf I London I Paris I Ho Chi Minh City