# SOA
# Zwischen Anspruch und Wirklichkeit

**Nicolai M. Josuttis**
**IT-communication.com**

Stand: 03/06

## SOA Prediction 1/2

- **According to Gartner:**
  - SOA will become the dominant framework for creating and delivering software
  - By 2006,
    - more than 60% of the IT professional services market will be based on the exploitation of Web services standards and technology
  - By 2008,
    - SOA will provide the basis for 80% of development projects
    - most application software revenue will come from products that were built using SOA
  - By 2010,
    - 80 percent of application software revenue growth, including licenses and subscription fees, will come from products based on SOA

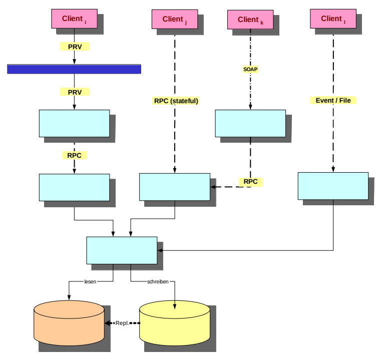
**[Gartner publication G00125868 from 12 May 2005]**

## SOA Prediction 2/2

- **According to Grady Booch, March 2006:**
  - „My take on the whole SOA scene is a bit edgier than most that I've seen. **Too much of the press about SOA makes it look like it's the best thing since punched cards.** ...
    Furthermore, if you follow many of these pitches, **it appears that you can do so with hardly any pain**: just scrape your existing assets, plant services here, there, and yonder, wire them together and suddenly you'll be virtualized, automatized, and servicized. **What rubbish**."
  - **„There are places where SOA is suitable, and places where it is not."**
  - **„SOA is a useful but insufficient mechanism for architectural decomposition."**
  - **„I've seen some folks suggest creating an SOA from the bottom up**: look at a silo, identify the potential services, and publish them, then weave a system together from them. This is in essence technology first. **In my experience, this is a recipe for disaster and/or serious over-engineering."**

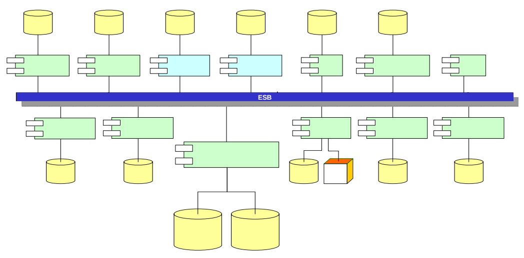
**[http://www.booch.com/architecture/blog.jsp, March 11, 2006]**

## SOA as „Brave New World"



It seems:
– with only a service bus
– all complexity is gone

WRONG!



## SOA: Vom Anspruch zur Wirklichkeit

### SOA = Infrastructure + Architecture

- **SOA = Infrastructure + Architecture**
- **Infrastructure**
  - Highly interoperable Enterprise Service Bus
- **Architecture**
  - Clear roles and responsibilities
  - Service classification
  - Patterns
- **For large systems**
  - Loose coupling
  - Different ownership boundaries

## SOA: Vom Anspruch zur Wirklichkeit

### Loose Coupling

- **Based on [Krafzig04]:**

| | **Tight Coupling** | **Loose Coupling** |
|---|---|---|
| **physical** | point-to-point | intermediator |
| **communication style** | synchronous | asynchronous |
| **type system** | strong | weak |
| **interaction pattern** | navigate through complex object trees | data-centric, self-contained message |
| **control of process logic** | central control | distributed control |
| **binding** | statically | dynamically |
| **platform** | strong dependencies | platform independent |
| **transactionality** | 2PC (two-phase commit) | compensation |

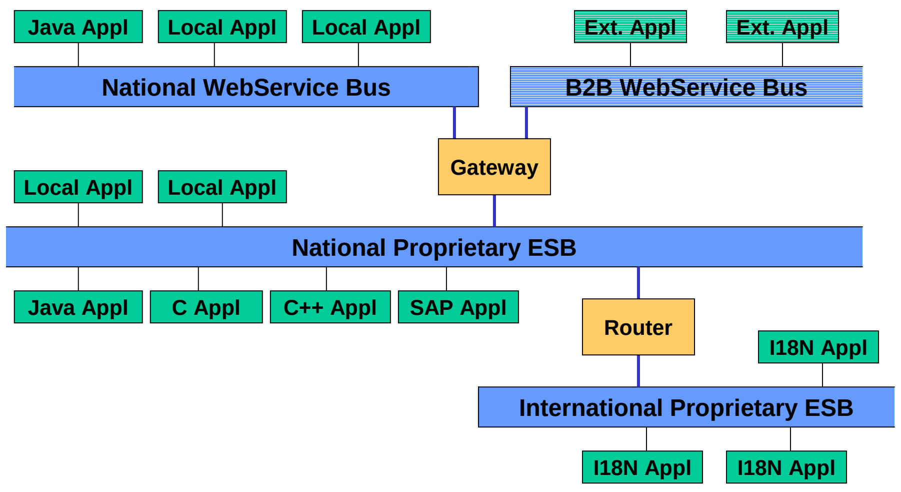
- **Note that „loose“ sounds simpler,
but is harder to implement and increases complexity**

## SOA: Vom Anspruch zur Wirklichkeit

7

josuttis | eckstein
IT communication

### Example Infrastructure



## SOA: Vom Anspruch zur Wirklichkeit

8

josuttis | eckstein
IT communication

### Service Classification

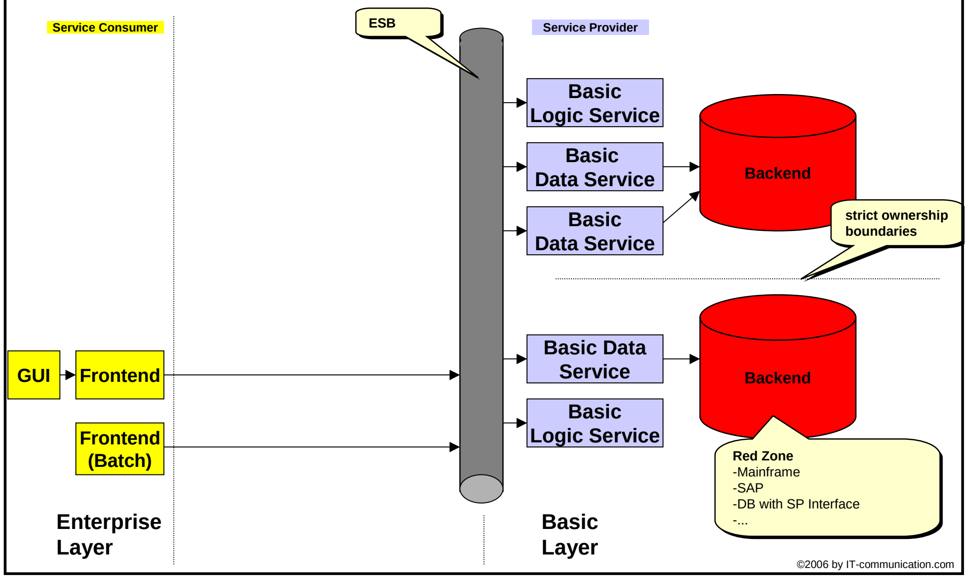
**According to [Krafzig04]:**

- **Basic Services**
  - data centric or logic centric
- **Intermediary Services**
  - composition of services without a state
- **Process Centric Services**
  - technical representation for/of business processes
  - stateful
- **Public Enterprise Services**
  - external enterprise interface
  - additional requirements for security, robustness, stability

## SOA: Vom Anspruch zur Wirklichkeit

9

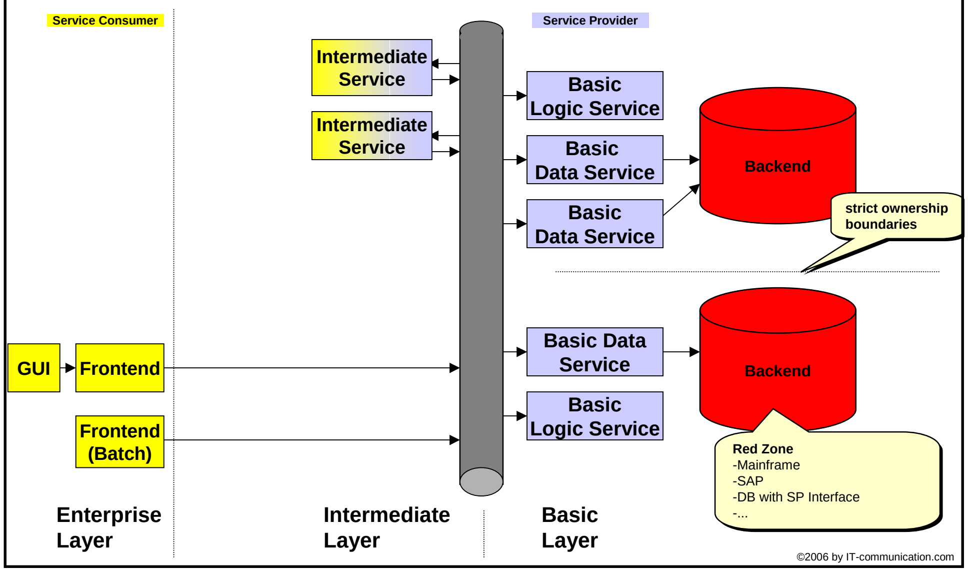
### Fundamental SOA



## SOA: Vom Anspruch zur Wirklichkeit

10

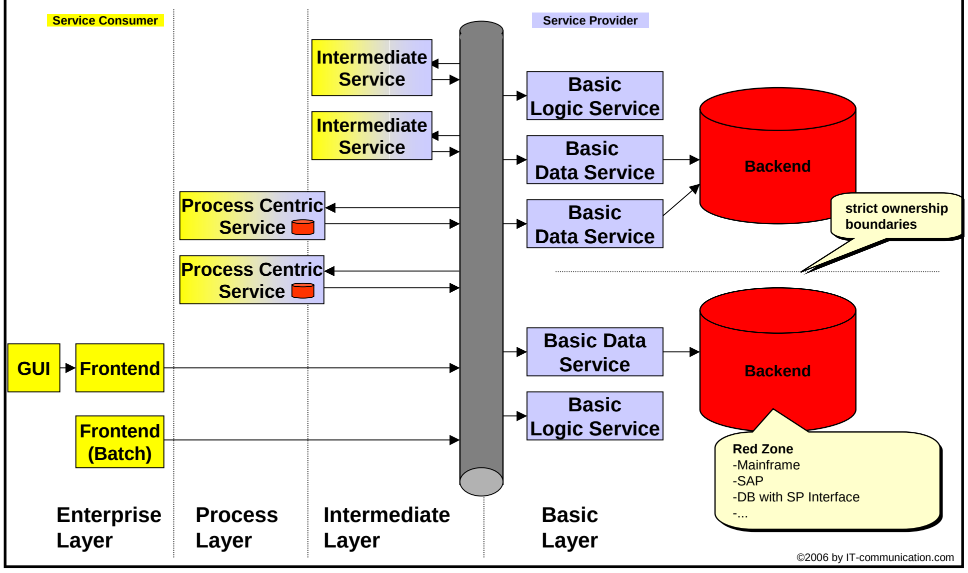
### Networked SOA



## SOA: Vom Anspruch zur Wirklichkeit
11

josuttis | eckstein
IT communication

### Process-enabled SOA



## SOA: Vom Anspruch zur Wirklichkeit
12

josuttis | eckstein
IT communication

### Managed SOA

- **Infrastructure to**
  - register and find services
  - manage dependencies between services
  - manage security issues
  - manage reuse of services and types
  - manage and verify service design patterns
  - find inconsistencies
  - verify service signatures
  - monitor performance
  - ...
- **Obvious solution:**
  - Central repository for all services
  - Validation of services calls inside the ESB

## Services and Performance

- **Performance is an issue**
  - at least, if an end user is waiting
- **Performance depends on**
  - Service provider (to provide data)
  - Infrastructure (to transfer data)
  - Service user (to process data)
- **Usually, performance is fine for trivial scenarios**
- **BUT:**
  - Reuse affects performance
  - Maintenance affects performance

## SOA Performance Corollary

**In performance critical SOA environments:**

a) **Service extensions are often modifications**
   - because additional data affects the running time

b) **Service segmentation depends** on
   - the architecture of the service consumer
   - the architecture of the service provider
   - the architecture of the service infrastructure

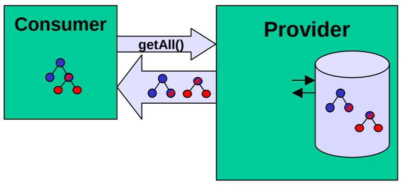
c) **The concept of reuse does not work as expected**
   - business case might not be reached

## SOA: Vom Anspruch zur Wirklichkeit

josuttis | eckstein
IT communication

### Example: Customer Call in Call Center

- **Loading all customer data at once took too long:**



**Note:**

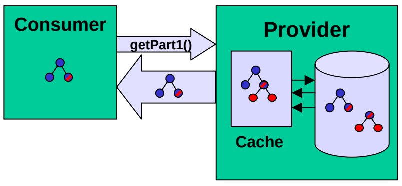
- **The separation of core and remaining customer data depends on business rules**
- **which might change**
- **But then**
  - It's *not* an easy new composition/orchestration
  - It's a service modification (or new service)

## SOA: Vom Anspruch zur Wirklichkeit

josuttis | eckstein
IT communication

### Example: Customer Call in Call Center

- **Loading all customer data at once took too long**
- **Solution:**
  - 1st Call:
    - returns core customer data and
    - initiates the load of all data into a cache



**Note:**

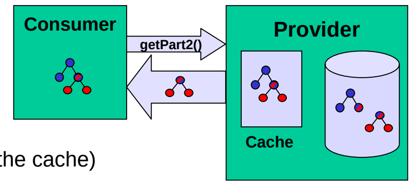
- **The separation of core and remaining customer data depends on business rules**
- **which might change**
- **But then**
  - It's *not* an easy new composition/orchestration
  - It's a service modification (or new service)

## SOA: Vom Anspruch zur Wirklichkeit

josuttis | eckstein
IT communication

### Example: Customer Call in Call Center

- **Loading all customer data at once took too long**
- **Solution:**
  - 1st Call:
    - returns core customer data and
    - initiates the load of all data into a cache
  - 2nd Call:
    - loads the remaining customer data (out of the cache)



**Note:**

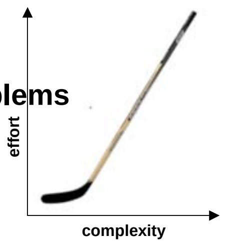
- **The separation of core and remaining customer data depends on business rules**
- **which might change**
- **But then**
  - It's *not* an easy new composition/orchestration
  - It's a service modification (or new service)

## SOA: Vom Anspruch zur Wirklichkeit

josuttis | eckstein
IT communication

### Tool Evolution

- **1st Generation: Retrofit**
  - old model with new syntax/technology
  - „Lipstick on a pig“
- **2nd Generation: Simple tools for simple problems**
  - correct model with but simplifications
  - „Hockey Stick Function“



- **3rd Generation: Efficient tools for complex problems**
  - understand how experienced developers really work
  - need experience of 2nd generation tools

**based on: Gregor Hohpe, Google @OOP2006**

## SOA: Vom Anspruch zur Wirklichkeit

### SOA Tooling

- **SOA is young**
- **We are between 1st and 2nd generation of tools**
  - beware of „Hockey stick function“
- **Note:**
  - Good tools can only solve problems we had in real life
  - IntelliJ and Eclipse came more than 5 years after Java was born

## SOA: Vom Anspruch zur Wirklichkeit

### Model Driven SOA

- **SOA forces MDSD:**
  - For simple small service specifications
    there is large and complex source code necessary

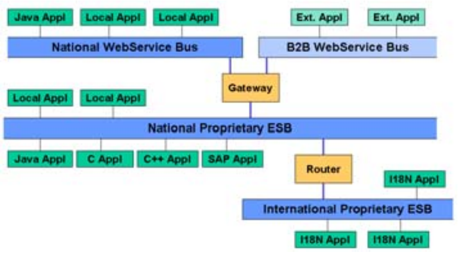
**Therefore:**

- **Generate the source code
  required by the SOA infrastructure
  from the individual service descriptions**

## Multiple Models and Tools

**In practice, we have**

- **Different modeling layers / models:**
  - Service interfaces (signatures)
  - Service implementations (source code)
  - Service dependencies
  - Quality of Service attributes
  - Deployment
  - Process modeling (BPEL etc.)
- **Different tools:**
  - Text editors
  - Excel
  - UML tools
  - XML editors
  - BPEL IDEs



## Models as Source (Code)

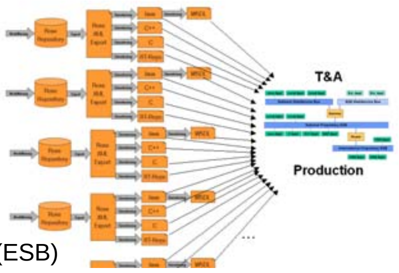
- **Models and tools must meet the usual requirements of source (code):**
  - Usual tool requirements
    - start within seconds
    - appropriate HMI
  - Usual configuration management requirement
    - versionable
    - support diffs
    - support merging
    - navigable
    - debugable
  - even with „protected zones“

josuttis | eckstein
IT communication

## Model Driven Processes

- **„Model Driven“ needs a working process**
- **The process has to define:**
  - Who defines when and where which aspect of the model?
  - When gets which aspect of the model processed?

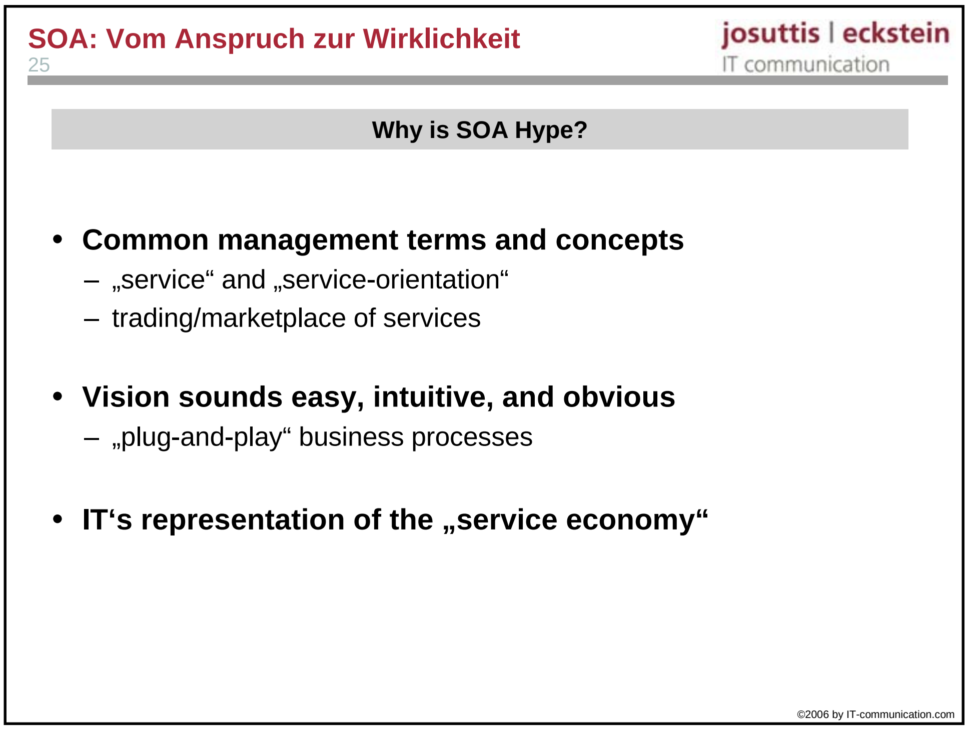
- **For large teams:**
  - no bottlenecks
    - no „model master“
    - no „BOM“ (Business Object Model)
      - OO does not scale
    - no central repository?
    - no verification inside the infrastructure (ESB)
    - support offline development (shortcuts)
  - understandable



SIGS DATACOM
Fachinformationen für IT-Professionals

josuttis | eckstein
IT communication

## Summary



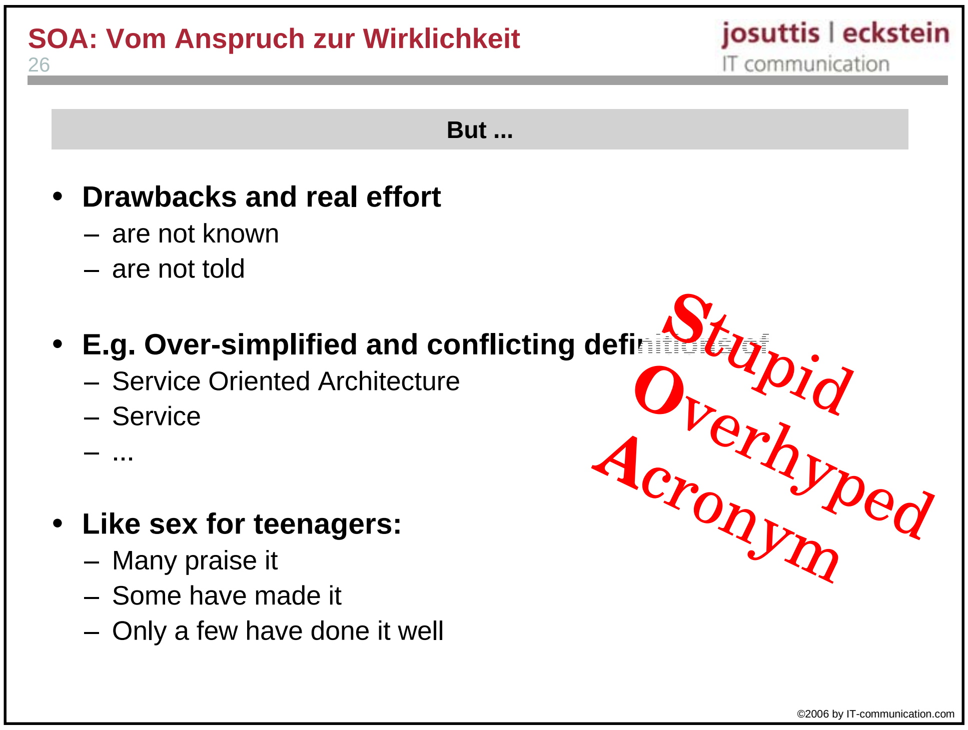
SOA: Vom Anspruch zur Wirklichkeit

25

josuttis | eckstein
IT communication

## Why is SOA Hype?

- **Common management terms and concepts**
  - „service“ and „service-orientation“
  - trading/marketplace of services
- **Vision sounds easy, intuitive, and obvious**
  - „plug-and-play“ business processes
- **IT‘s representation of the „service economy“**



SOA: Vom Anspruch zur Wirklichkeit

26

josuttis | eckstein
IT communication

## But ...

- **Drawbacks and real effort**
  - are not known
  - are not told
- **E.g. Over-simplified and conflicting definitions of**
  - Service Oriented Architecture
  - Service
  - ...
- **Like sex for teenagers:**
  - Many praise it
  - Some have made it
  - Only a few have done it well

Stupid
Overhyped
Acronym

## SOA: Vom Anspruch zur Wirklichkeit

josuttis | eckstein
IT communication

### Step 1: Understand SOA

- **SOA is an architectural style**
  - Shared architectural vision is key
- **Think big**
  - SOA is a concept for large decoupled systems
- **SOA should accept heterogeneity**
  - different languages
  - different platforms
  - different middleware
  - ...
- **While agility accepts that requirements are never stable you might consider SOA as an approach that accepts that systems are never harmonized**

## SOA: Vom Anspruch zur Wirklichkeit

josuttis | eckstein
IT communication

### Step 2: Establishing a SOA

- **Kent Beck:**
  - *„Start stupid and evolve“*
- **Start with fundamental SOA**
  - establish **infrastructure and architecture** for
    - a small number of services
    - a small number of service participants
- **Grow**
  - increase number of services and participants
  - introduce Networked SOA, Process-enabled SOA
  - introduce Managed SOA
- **Plan time to refactor**

josuttis | eckstein
IT communication

## SOA Summary

- **SOA = Infrastructure + Architecture**
- **Individual service design is tough**
- **There are tradeoffs between**
  - performance, reusability, and scalability
- **Never introduce bottlenecks for the service development**
- **Too many Standards but not enough Tooling**
- **You need**
  - architectes with experience, courage, and who serve
  - time to learn, review, and refactor

josuttis | eckstein
IT communication

## So Long, and Thanks for All the Fish

**Nicolai Josuttis**

**www.it-communication.com**
**josuttis@it-communication.com**

Gaussstr. 29
D - 38106 Braunschweig

Tel.: +49 531 / 129 88 86
+49 700 / 5678 8888
+49 700 / JOSUTTIS