

# 1. Introduction

Jiří Vokřínek

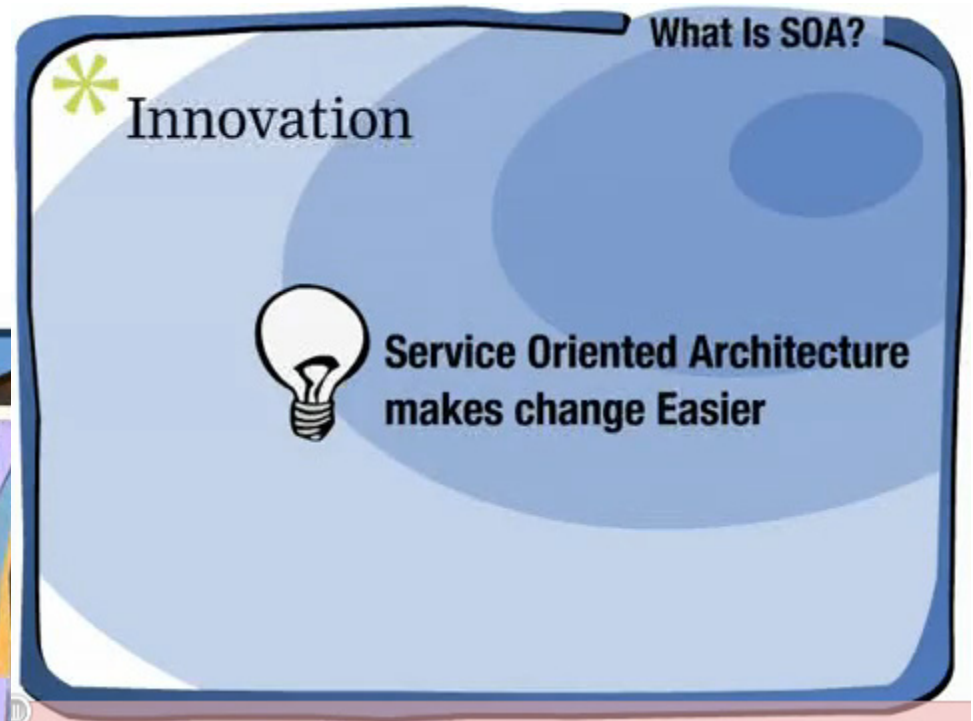
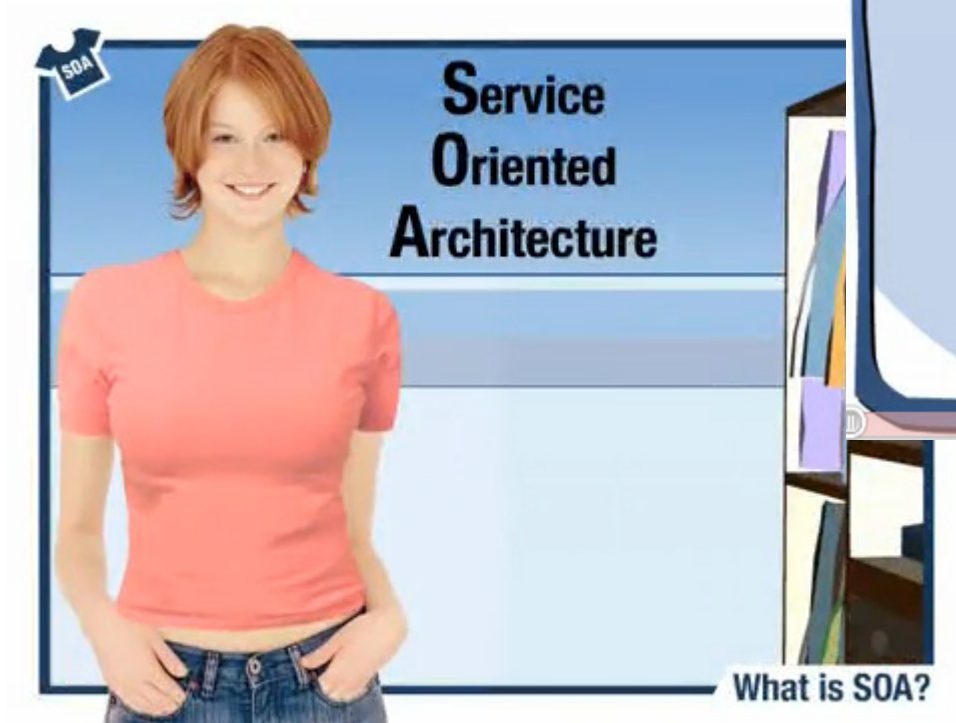
Agent Technology Center  
Department of Cybernetics

Faculty of Electrical Engineering, Czech Technical University in Prague

[vokrinek@agents.felk.cvut.cz](mailto:vokrinek@agents.felk.cvut.cz)

<http://agent.felk.cvut.cz>

# YouTube What is Service Oriented Architecture SOA?



[http://www.youtube.com/watch?v=sbd\\_1G8Kqjs&feature=channel](http://www.youtube.com/watch?v=sbd_1G8Kqjs&feature=channel)  
<http://www.youtube.com/watch?v=dyHWAiG6c-Y&feature=channel>

Ibangz

3 months ago

Thanks for the explanation! My lecture notes didn't describe it as great as this.

MississippiUS

11 months ago

Excellent explanation of SOA,  
So basically it is what n-tier is for Web applications.

WaffleTroll

2 years ago

The current implementations of SOA using BPEL, XML, SOAP..etc are immature pathetic jokes. They lack essential primitives such as distributed transactions ("Compensation" does not count), security primitives, common logging and monitoring all non-existent.

SOA is far from a new concept. LoD has been around for decades and all well designed distributed systems have implemented these concepts from the very beginning.

SOA in its now popular form promises the world and delivers a headache.

[codematrix](#)

2 years ago

I hate when architects say that they build some services using Web Services and then say that they that's SOA. Web Services != SOA. SOA is a methodology/approach.

[pdasaro](#)

1 month ago

SOA or SOS? Wow, were can I buy SOA? Who makes it? Is it compatible in my multi platformed environment? This is just bullshit for non technical project managers! We still have to buy hardware , software and network components and configure them. There is no magical product that does the hard stuff for you! This is just another name for the same old shit.

# Service-oriented Architecture

- is a flexible **set of design principles** used during the phases of systems development and integration
- provide a **loosely-integrated** suite of services that can be used within multiple business domains
- defines how to integrate widely disparate applications for a world that is **distributed** and uses **multiple** implementation **platforms**

# Service-oriented Architecture

- defines the **interface** in terms of protocols and functionality
- requires **loose coupling** of services with operating systems, and other technologies that underlie applications
- separates functions into **distinct** units, or **services**, which developers make accessible over a network

# Services

- allow users to **combine and reuse** them in the production of applications
- services and their corresponding consumers **communicate** with each other by passing data in a **well-defined**, shared **format**, or by **coordinating an activity** between two or more services

# Services

- implementations rely on a **mesh** of software services
- services comprise unassociated, loosely coupled **units of functionality** that have no calls to each other embedded in them
- each service implements **one action**
- instead of services embedding calls to each other in their source code they use **defined protocols** that describe how services pass and parse messages, using **description metadata**



# Principles

- reuse, granularity, modularity, composability, componentization and interoperability
- standards compliance (both common and industry-specific)
- services identification and categorization, provisioning and delivery, and monitoring and tracking

# Principles

- service **encapsulation** – many services are consolidated for use under the SOA; often such services were not planned to be under SOA
- service **loose coupling** – services maintain a relationship that minimizes dependencies and only requires that they maintain an awareness of each other
- service **contract** – services adhere to a communications agreement, as defined collectively by one or more service-description documents

# Principles

- service **abstraction** – beyond descriptions in the service contract, services hide logic from the outside world
- service **reusability** – logic is divided into services with the intention of promoting reuse
- service **composability** – collections of services can be coordinated and assembled to form composite services
- service **autonomy** – services have control over the logic they encapsulate

# Additional Constraints

- **stateless service** – each message that a consumer sends to a provider must contain all necessary information for the provider to process it. This is effectively "service in mass production" since each request can be treated as generic. There are **no intermediate states** to worry about, so recovery from partial failure is also relatively easy. This makes a service more reliable.

# Additional Constraints

- **stateful service** – holds a **session** between a consumer and a provider. Stateful services require both the consumer and the provider to **share the** same consumer-specific **context**. The drawback of this constraint is that it may **reduce the** overall **scalability** of the service provider because it may need to remember the shared context for each consumer.

# Additional Constraints

- **idempotent request** – duplicate requests received by a service have the same effects as a unique request. This constraint allows providers and consumers to improve the overall service reliability by simply repeating the request if faults are encountered.



# **SOA:** **an example** **Service-oriented** **architecture**

Darmstadt University of Applied Sciences  
Prof. Dr. Bernhard Humm



# The (fictitious) company: Christopher Columbus Travel Pty Ltd.

---

## ■ Products:

- Package holidays, i.e. transport + accommodation
- Short-distance (Germany), medium-distance (e.g., Mallorca), long-distance (e.g., south east Asia)

## ■ Customers:

- Private customers
- From low-budget to premium
- In various European countries
- Diverse brands, focused on customer segments

## ■ Company:


- Multi-national tour operator
- Several thousand travel agencies
- Large IT department





# Strategic business decision in order to differentiate from competitors and to develop new markets

---



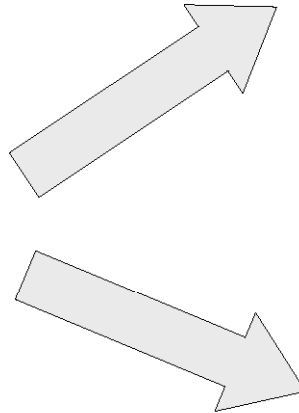
In addition to the classic package holiday, new custom holidays shall be offered

# The strategic decision has multiple implications

---

## Strategic decision

In addition to the classic package holiday, new custom holidays shall be offered



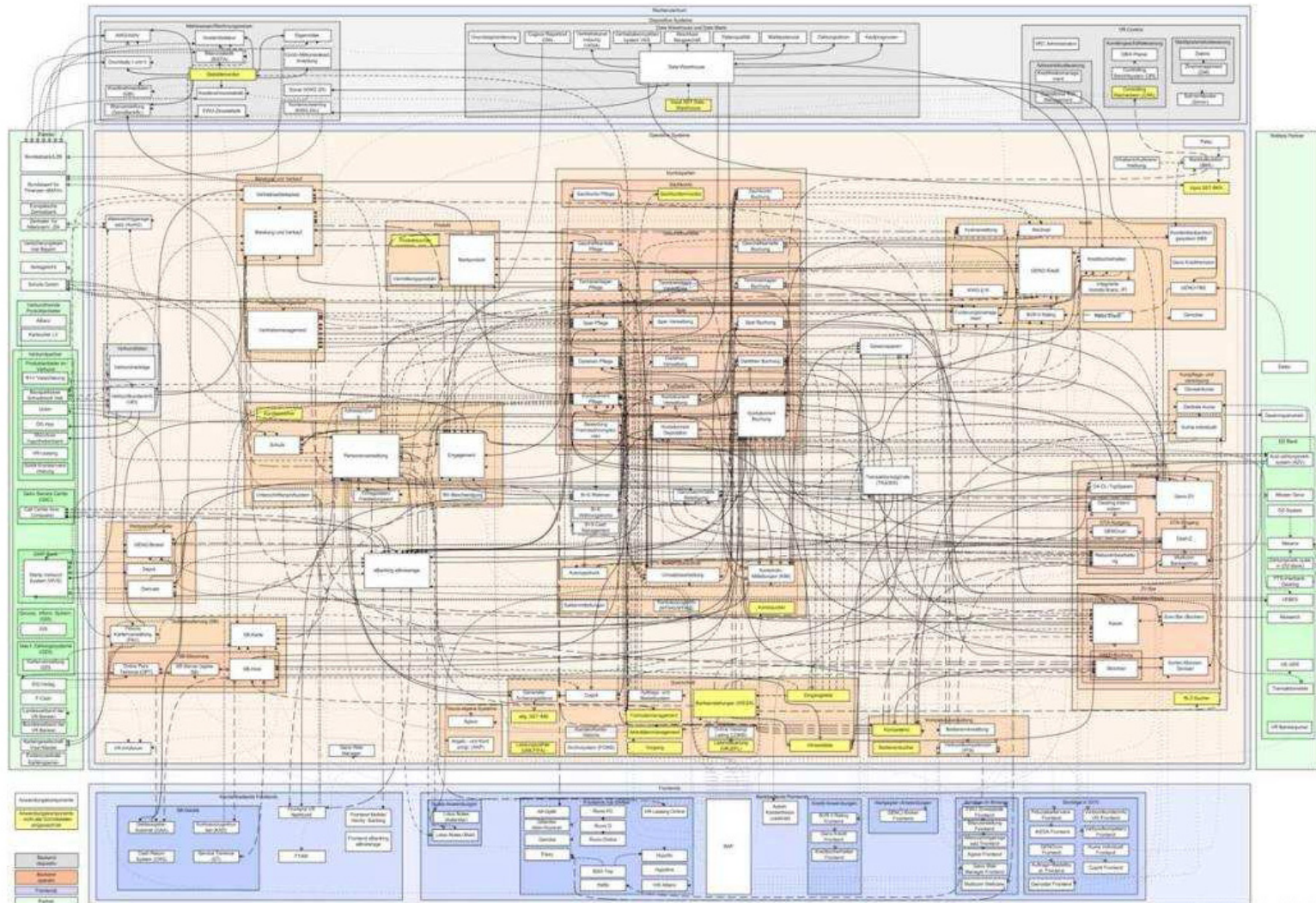
## Business implications

- New business partners
- Marketing for new product
- Adapted sales channels (Internet)
- Modified operations
- ...

## IT implications

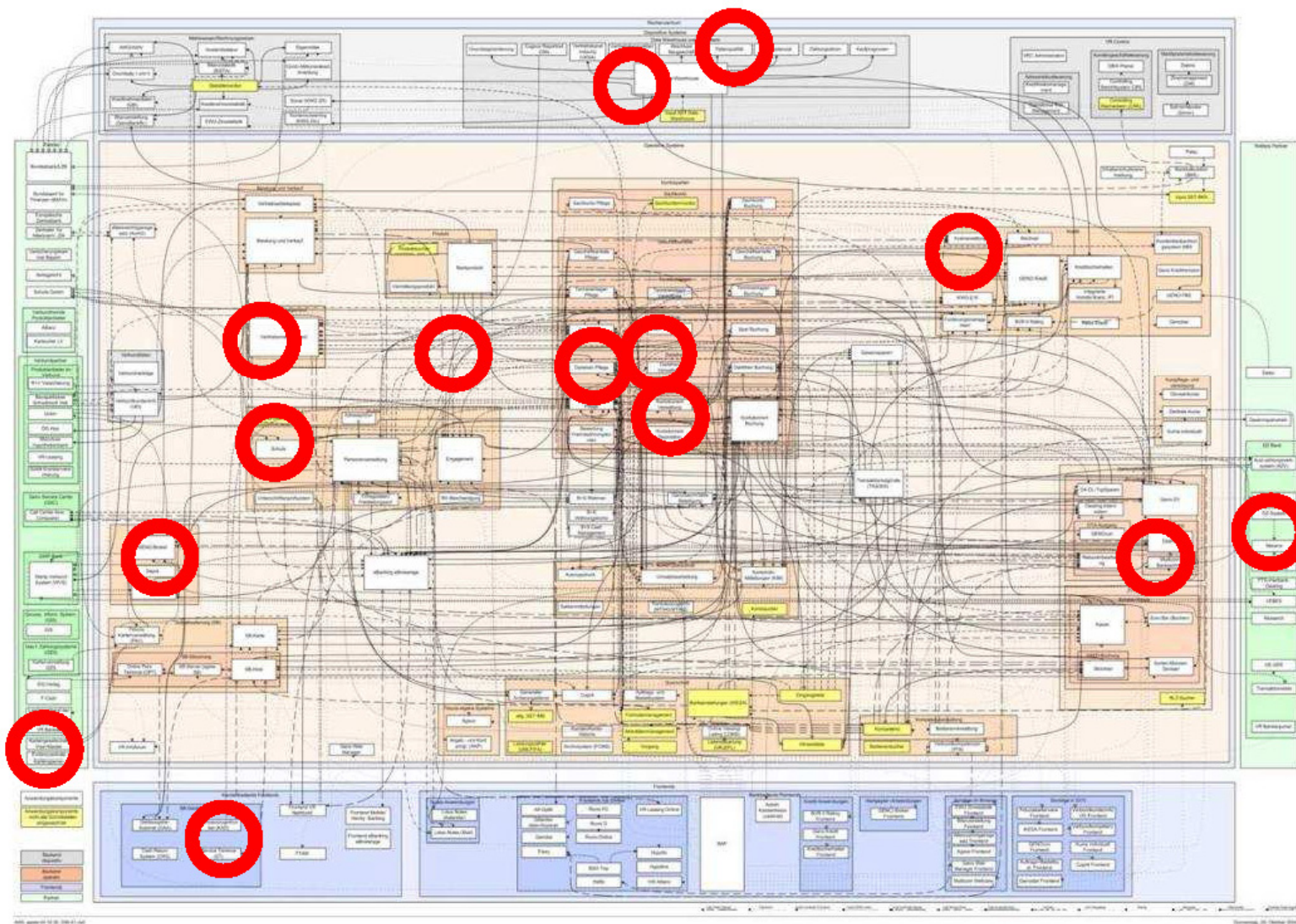
- New back-end applications
- New Internet portals
- Integration of new applications in legacy applications
- ...

# The existing enterprise IT architecture is complex and has grown over 20 years

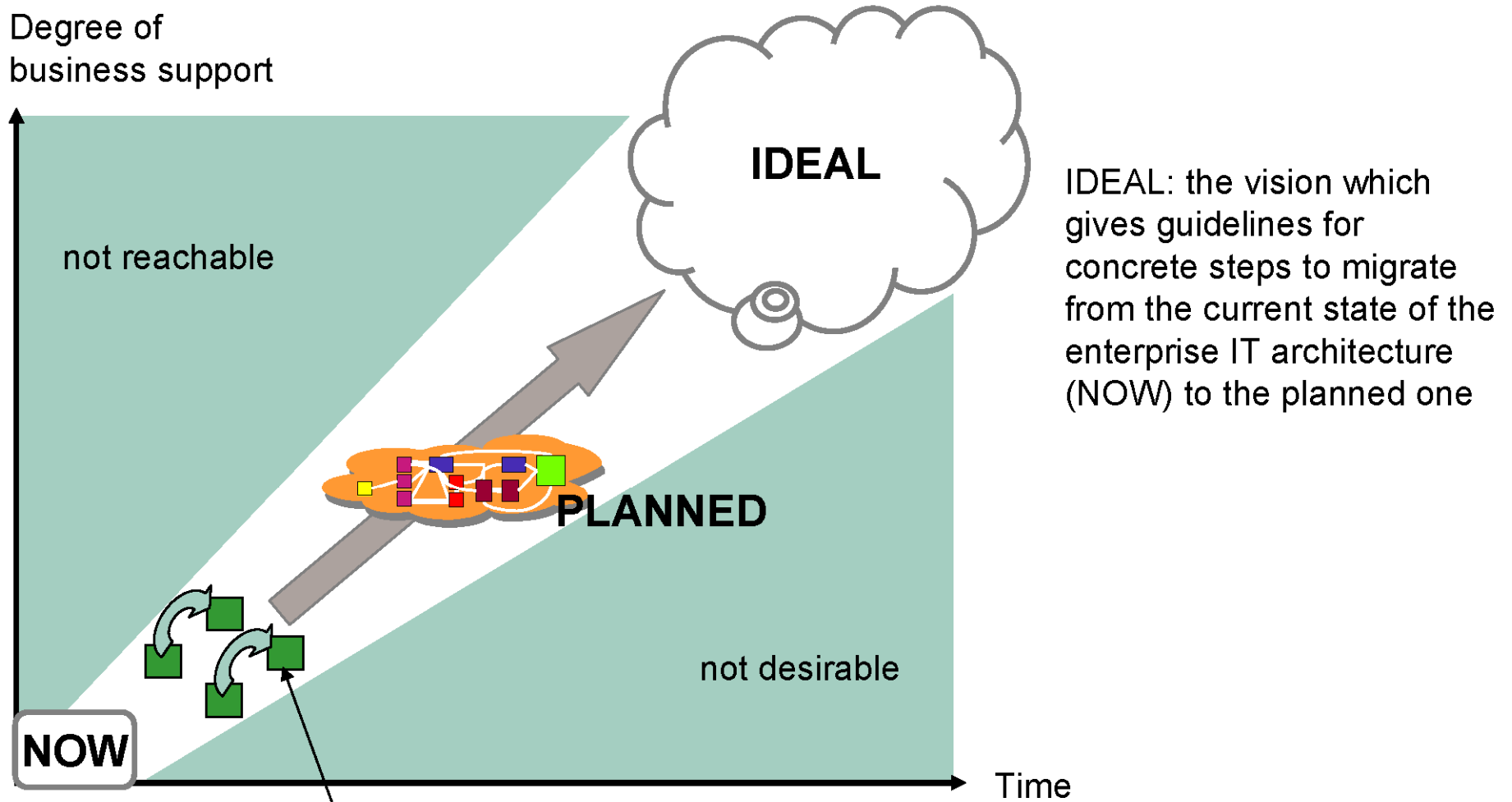




# The strategic decision has implications on many parts of the enterprise IT architecture

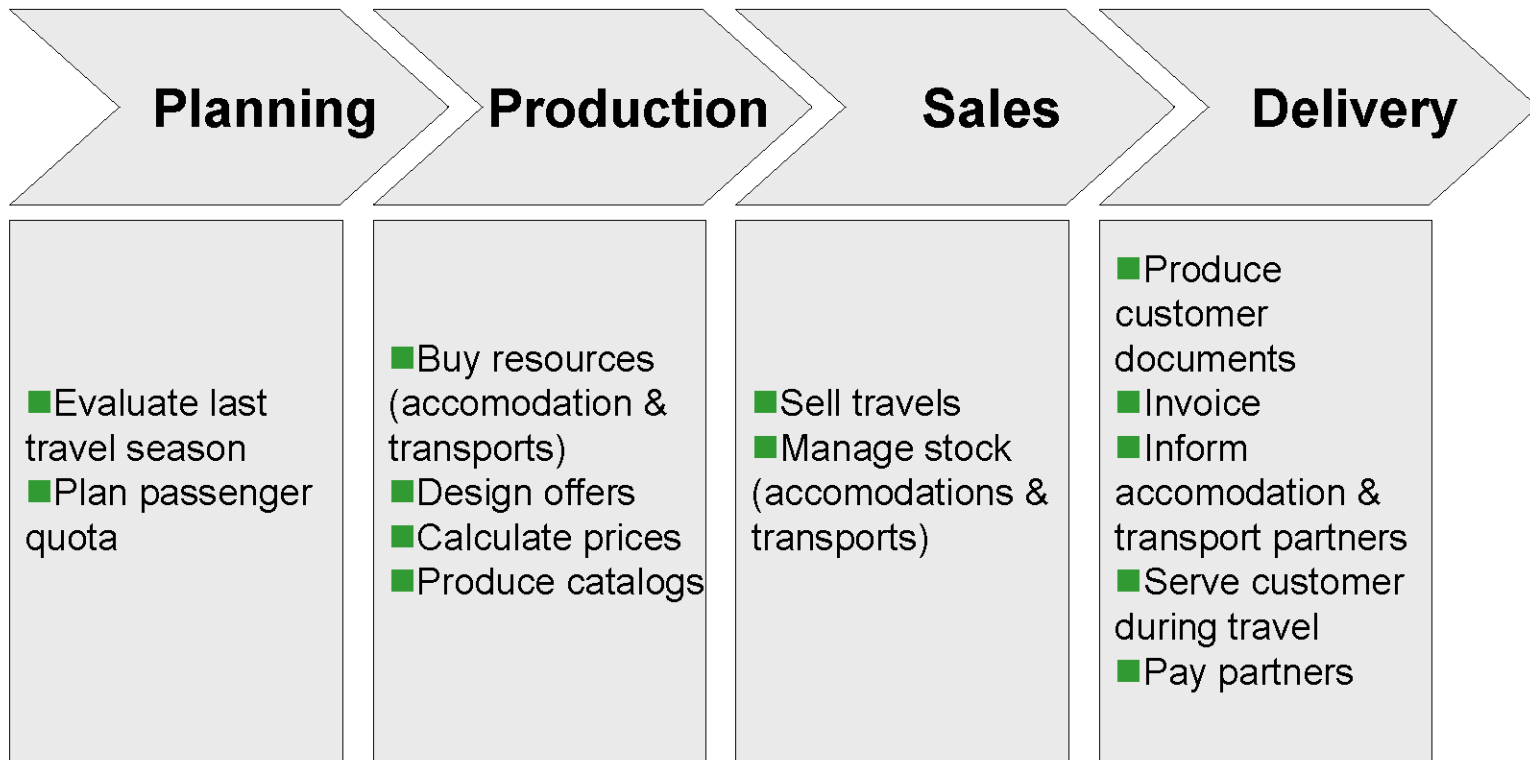


# The massive changes of the enterprise IT architecture necessitate a re-structuring



# Step 1: Identify core business processes and services

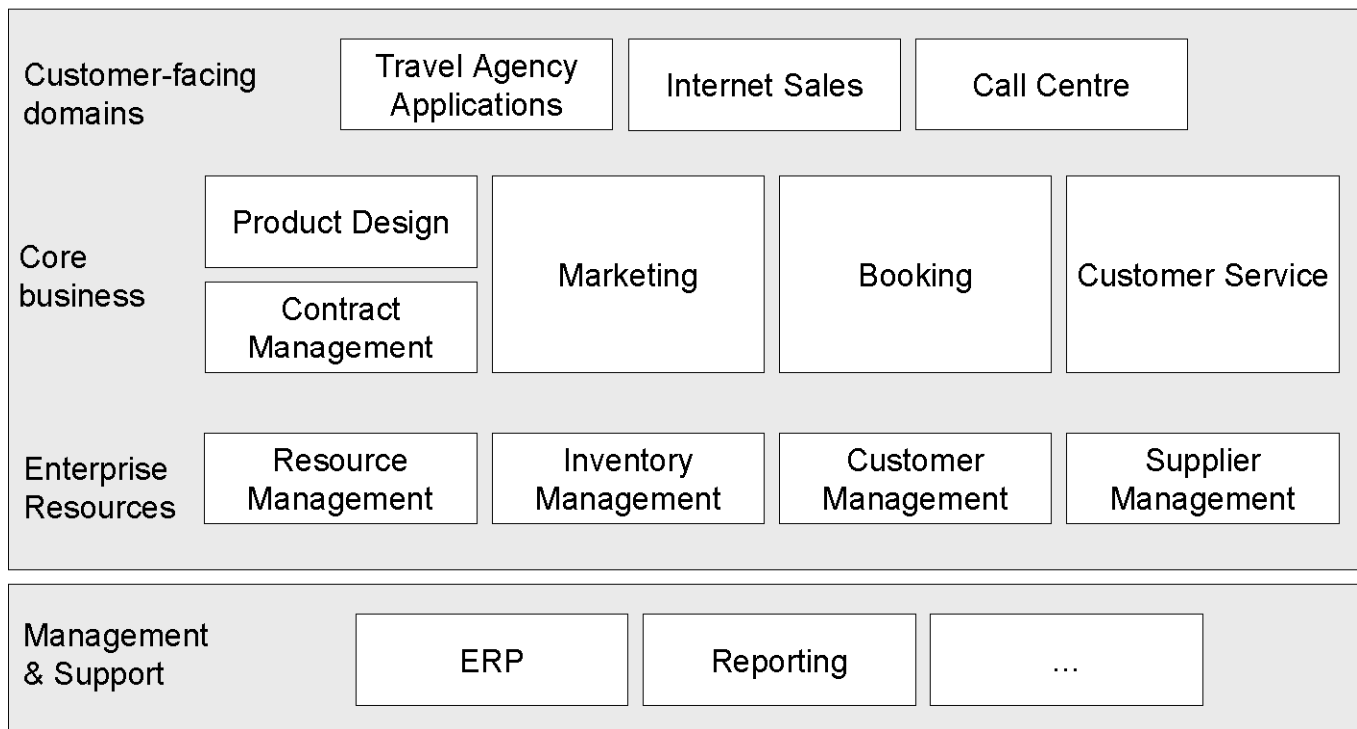
---



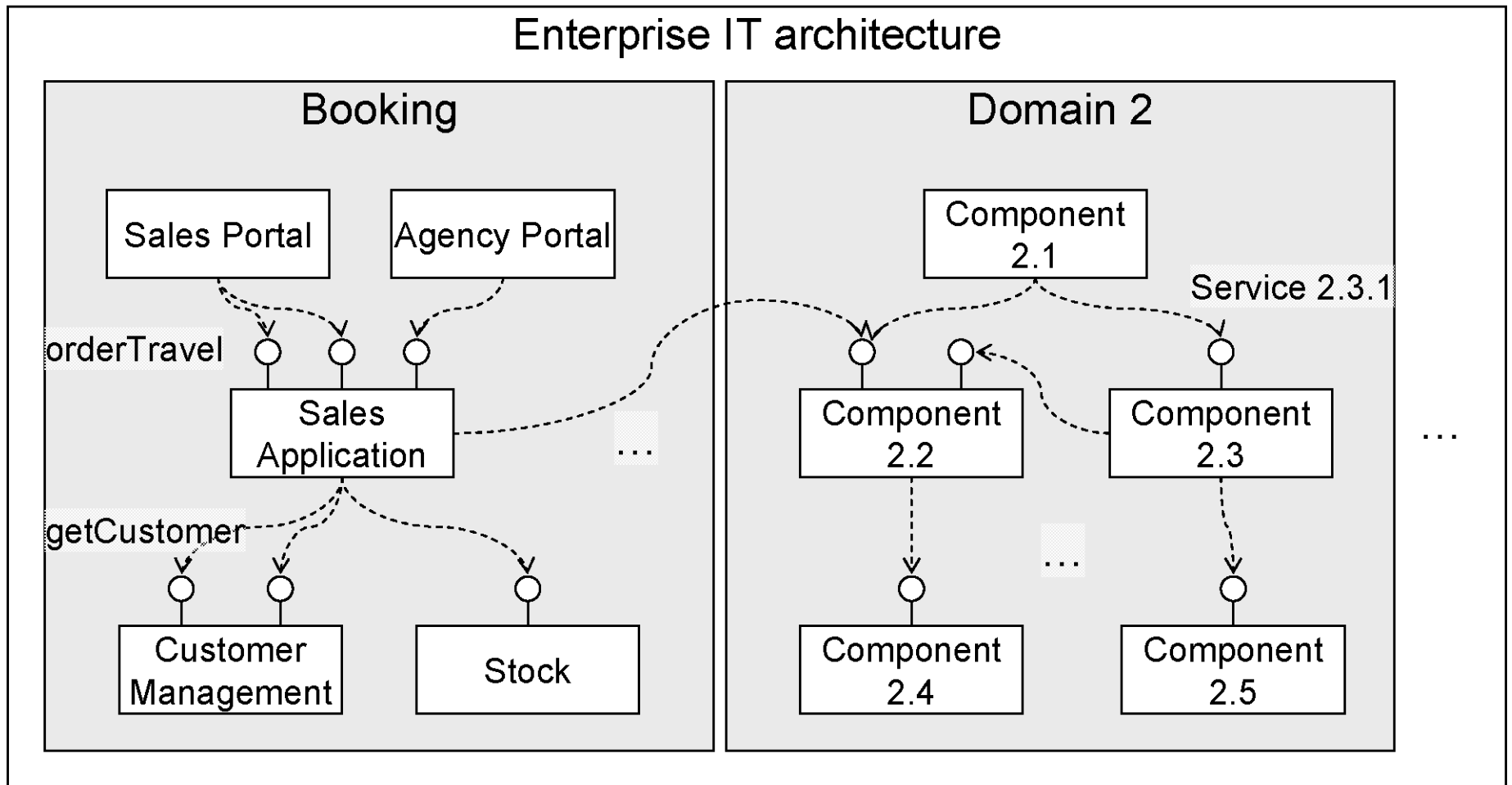
## Step 2: Identify domain model

---

- The domain model forms the background layer of the enterprise IT architecture diagram.
- Structuring the enterprise IT architecture according to the planned domain model allows you to identify necessary changes.



## Step 3: Identify components and services





# Step 4: Implement

- Design & implement components and interfaces
- Provide interfaces for existing components
- Integrate
- Test
- Put into Operation

