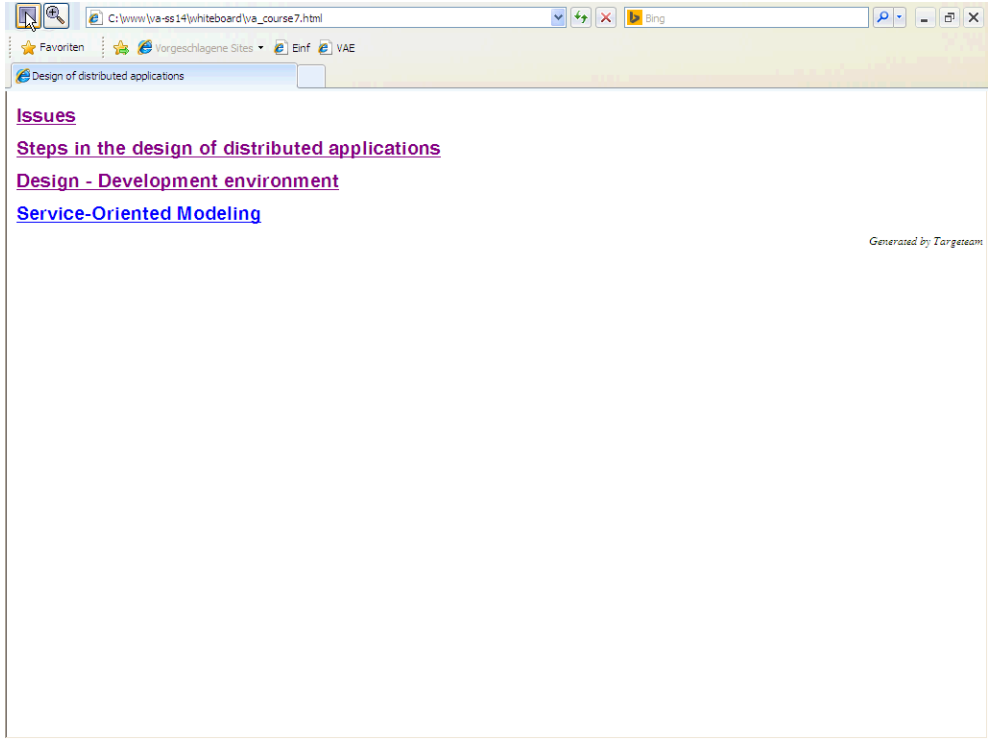
Script **generated by TTT**

Title: Distributed_Applications (30.06.2014)

Date: Mon Jun 30 09:18:30 CEST 2014

Duration: 43:12 min

Pages: 14



Service-Oriented Modeling

Ideas and proposals emerged to transfer the service approach to the design and modeling of software systems.

Definition: Service-oriented modeling (SOM) is the discipline of modeling business and systems, for the purpose of designing and specifying service-oriented business systems within SOA.

> create models that provide a comprehensive view for the analysis, design, and architecture of all software components in an organization.
>
> envision the coexistence of services in an interoperable computing environment.

Definition: The **service-oriented modeling framework** (SOMF) is a service-oriented development life cycle methodology that provides practices, disciplines and a universal language to provide tactical and strategic solutions to enterprise problems

Service Evolution

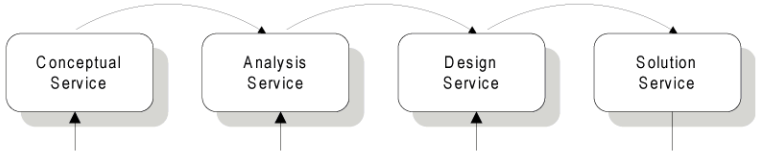
Life Cycle Structure

Life Cycle Modeling

SOM Framework

Service Evolution

SOM advocates the transformation of a service through 4 states.



1. conceptual service: in its inception, a service appears merely as an idea or concept.
2. analysis service: it becomes a unit of analysis.
3. design service: it evolves into a design entity.
4. solution service: it ends in a physical solution that is ready to be deployed in the production environment.

identifies the elements for service development and operations. It consists of 4 major components.

timeline : defines the life span of a service.

events : 2 types of events during the service life span.

predicted and scheduled events, e.g. milestone, planning stage or deployment stage.

unexpected events, e.g. stock market crash, trading volume exceeds capacity of trading service.

events have beginnings and may last for a while.

seasons : services live through 2 major life cycle seasons.

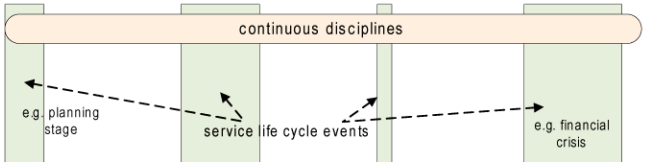
design-time season: services are conceptualized, analyzed, designed, constructed and tested.

run-time season: services are managed, monitored, and controlled to ensure proper performance.

disciplines : identify modeling and nonmodeling best practices and standards to be pursued throughout the service life cycle.

season disciplines: e.g. service-oriented conceptualization, business integration or construction.

continuous disciplines: e.g. service portfolio management, service governance.



unexpected events, e.g. stock market crash, trading volume exceeds capacity of trading service.

events have beginnings and may last for a while.

seasons : services live through 2 major life cycle seasons.

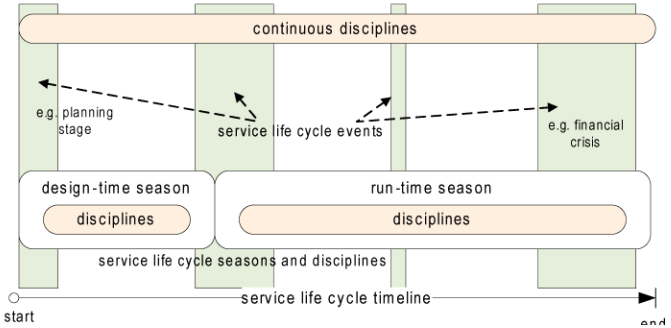
design-time season: services are conceptualized, analyzed, designed, constructed and tested.

run-time season: services are managed, monitored, and controlled to ensure proper performance.

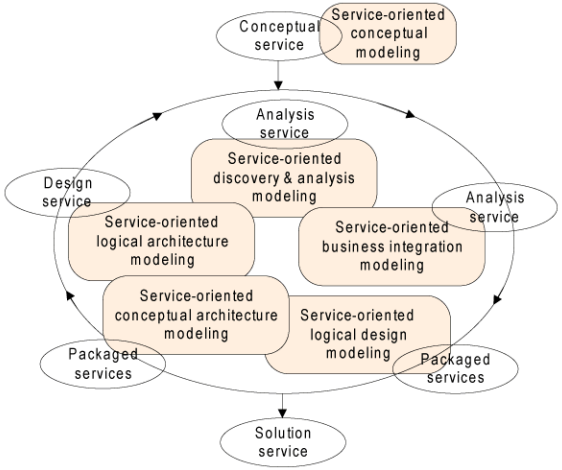
disciplines : identify modeling and nonmodeling best practices and standards to be pursued throughout the service life cycle.

season disciplines: e.g. service-oriented conceptualization, business integration or construction.

continuous disciplines: e.g. service portfolio management, service governance.



The following core processes can be identified in which business and IT personnel must be engaged to produce design and solution artifacts.

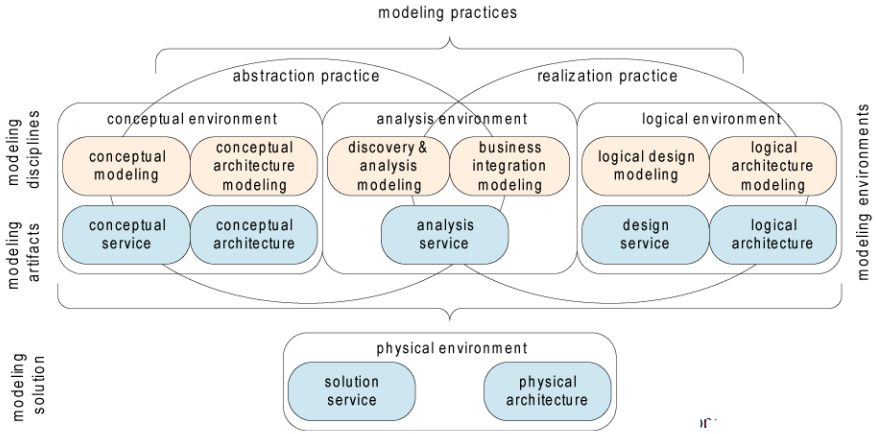


Conceptual modeling : identify driving concepts behind future solution services.

Discovery & analysis modeling : discover and analyze services for granularity, reusability, interoperability, loose-coupling, and identify consolidation opportunities for the existing software assets.

Business integration modeling : identify service integration and alignment opportunities with business

Modeling components and disciplines are integrated into a SOM framework.



Design of distributed applications

Distributed file service

Issues

This section introduces schemes for replication and concurrency control in the context of distributed file services.

What are the general characteristics of a distributed file service?

How to maintain consistency of replicated files?

What are voting schemes?

Presentation of the Coda file service.

Definitions

Definition: A **distributed file system** (e.g. Sun Network File System (NFS)) is characterized by:

a logical collection of files on different computers into a common file system, and

computers storing files are connected through a network.

Definition: A **distributed file service** is the set of services supported by a distributed file system. The services are provided by one or several file servers; a **file server** is the execution of file service software on a computer.

Definition: Allocation is the placement of files of a distributed file system on different computers.

Definition: Relocation changes file allocation within the distributed file system.

Definition: Replication

there exist multiple copies of the same file on several computers.

Replication degree REP_d of a file d: total number of copies of d within the distributed file system.

If replication transparency is supported, the user is unaware of whether a file is replicated or not.

Introduction

When a group of programmers has the task to build a distributed application, in addition to distributed code management there is also the need for distributed file services.

Two consistency types

In the context of replicated files we can distinguish between two types of file consistency.

Internal Consistency

A single file copy is internally consistent, e.g. by applying a "2-phase commit" protocol.

Mutual Consistency

It is obvious that all copies of replicated information should be identical ⇒ all file copies are mutually consistent, for example by applying the "multiple copy update" protocol.

Strict mutual consistency : after executing an operation, all copies have the same state.

Loose mutual consistency : all copies converge to the same consistent state of information.

Replica placement

A major issue of distributed data store is the decision when and where to place the file replicas.

Permanent replicas

The number and placement of replicas is decided in advance, e.g. mirroring of files at different sites.

Server-initiated replicas

They are intended to enhance the performance of the server.

Dynamic replication to reduce the load on a server.

file replicas migrate to a server placed in the proximity of clients that issue file requests.

Client-initiated replicas

Client-initiated replicas are more commonly known as caches.

Used only to improve access times to data.

Client caches are normally placed on the same machine as its client.

Replicas are only kept for a limited time.

Introduction

When a group of programmers has the task to build a distributed application, in addition to distributed code management there is also the need for distributed file services.