Title: Distributed Applications (09.07.2012)
Date: Mon Jul 09 09:15:40 CEST 2012
Duration: 46:35 min
Pages: 22

Design of distributed applications

Issues
Steps in the design of distributed applications
Design - Development environment
Service-Oriented Modeling

...continued

Integrated with ISO with the goal of defining a reference model for distributed applications
integrating a wide range of standards for distributed systems, e.g. ISO/OSI reference model
Reduction of complexity by specifying different levels of abstractions of the distributed system ("viewpoints")
Enterprise viewpoint: deals with the overall goals that the distributed system should reach within the organization
Information viewpoint: focuses on aspects of the structure, control of and the access to information
Computation viewpoint: aspects of the logical distribution of data and subsystems
Engineering viewpoint: physical distribution of data and subsystems
Technology viewpoint: the different physical and technical subsystems, e.g. network, hardware platforms.
Model Driven Architecture (MDA)

concept for structured and documented software development
OMG standard (Object Management Group)
use of architectural models

MDA Concept

Models
Definition: A model is a description of (part of) a system written in a well-defined language.
Definition: A well-defined language is a language with well-defined form (syntax), and meaning (semantics), which is suitable for automated interpretation by a computer

2. Step: mapping to PSM
mapping of PIM to platform specific models (PSMs)

- PSM: web service model
- PSM: Java/EJB model
- PSM: Corba model
- PSM: ...

- PSM models realization of software solution in UML
- Example: software components are Web services and communication via SOAP

3. Step: code generation

consists of 3 steps
- development of platform independent models (PIMs)
- mapping to platform dependent models (PSMs)
- implementation, integration and test

* transformation between models (PIM → PSM, PSM → code)

1. Step: development of PIM
2. Step: mapping to PSM
3. Step: code generation

Code generation: system development, test
AutoFocus is a platform to specify distributed systems developed by the group of Prof. Broy, TU München based on formal methods of systems engineering integrates hierarchical description techniques allows distributed and platform independent development project advanced to AutoFocus supporting the following functionality
Requirement analysis tool (AutoRAID), such as use-cases and scenarios, business and application requirements Design modelling views and editors, such as system structure diagram, state transition diagram, message sequence charts interactive simulation environment, code generation, consistency maintenance support.

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 Other SOA Design Methodologies

Design - Development environment
use of Software Engineering concepts, methods and tools to design and develop distributed applications software development cycle is divided into phases requirements analysis, specification, design, implementation, test and integration, maintenance for details see Software Engineering courses
Open Distributed Processing (ODP) Model Driven Architecture (MDA) AutoFocus

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 non-modeling 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.

---

**Life Cycle Structure**

- **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 non-modeling 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.

---

**Life Cycle Modeling**

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 domains' processes (organizations, IT, products, geographical locations).

**Local design modeling**: establish service relationships and message exchange paths; address service
Life Cycle Modeling

**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 domains’ processes (organizations, IT, products, geographical locations).

**Logical design modeling**: Establish service relationships and message exchange paths; address service visibility; prepare service logical compositions; model service transactions.

**Conceptual architecture modeling**: Establish an SOA architectural direction; select an SOA technological environment; establish an SOA technological stack; identify technological asset ownership.

**Logical architecture modeling**: Integrate SOA software assets; establish SOA logical environment dependencies; foster discoverability, loose coupling and interoperability.

SOM Framework

Modeling components and disciplines are integrated into a SOM framework.

Other SOA Design Methodologies

A brief overview of some other SOA design methodologies

**Creating Service-Oriented Architectures (CSOA) by Barry & Associates**

- Focus is on technical aspects
- Consist of the 5 phases
  - Design service
  - Discover & analysis modeling
  - Business integration modeling
  - Logical design modeling
  - Logical architecture modeling

**Service-Oriented Transformation of Legacy Systems (SOTLS) by Nadhan**

- Targets the stepwise evolution of existing application systems towards service-oriented architectures
- Focus is on technical aspects
- Incorporates the perspectives of the service provider as well as the service consumer

**Service-Oriented Design and Development (SOAD) by Papazoglou**

- Planning, analysis, service design, service construction, service test, service deployment/execution and service management/monitoring
Design of distributed applications

Issues
Steps in the design of distributed applications
Design - Development environment
Service-Oriented Modeling

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 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.