Analiza și proiectarea sistemelor software

curs 13

Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

PLAN CURS

Arhitecturi software orientate pe servicii

Servicii web clasice

Servicii Web RESTful

Ingineria serviciilor

Compunerea serviciilor

Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

SERVICII SOFTWARE

- SERVICIU = *“an act or performance offered by one party to another. Although the process may be tied to a physical product, the performance is essentially intangible and does not normally result in ownership of any of the factors of production”.*

- SERVICIU SOFTWARE = instanță a noțiunii de SERVICIU

Definiție (OASIS*) *:* a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description.

- *OASIS - Organization for the Advancement of Structured Information Standards*

Esența unui serviciu software constă în faptul că oferirea acestuia este independentă de utilizatorii lui.

Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

SERVICII – COMPONENTE REUTILIZABILE

DEFINIŢII

–***Serviciu software*** - componentă software slab cuplată, reutilizabilă, care încapsulează funcţionalitate discretă ce poate fi distribuită şi accesată programatic.

–***Serviciu Web*** – serviciu software accesat utilizând protocoale standard pentru Internet şi bazate pe XML.

Caracteristici distinctive ale serviciului faţă de componenta clasică:

•slab cuplat

•ascunde detaliile de platformă și de limbaj de programare

•nu are (în general) interfeţe de tip ‘requires’.

•comunicare exclusiv prin mesaje

•poate fi distribuit în Internet: mesajele sunt exprimate în XML şi transferate utilizând protocoale standard de transport în Internet (HTTP şi TCP/IP).

BENEFICIILE ABORDĂRII ORIENTATĂ PE SERVICII

- Integrare servicii de la furnizori multipli. Furnizorul de servicii poate fi extern sau intern firmei ce dezvoltă aplicația.

- Serviciul poate fi folosit de către orice utilizator autorizat, pe baza informațiilor publicate despre serviciu de către furnizor.

- Legarea serviciului la aplicație se face la instalare sau la execuție $\Rightarrow$ aplicațiile pot fi reactive și își pot adapta operarea la modificările din contextul de execuție.

- Este posibilă construirea oportunistă de noi servicii prin legare servicii existente în moduri inovative.

Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

BENEFICIILE ABORDĂRII ORIENTATĂ PE SERVICII

- Plata serviciului se poate face conform utilizării acestuia, nefiind necesare achiziții de componente utilizate rar.

- Dezvoltare aplicații de dimensiuni reduse (important pentru dispozitivele mobile cu resurse de procesare și de memorie reduse). Procesele cu calcule intensive vor fi realizate de servicii externe.

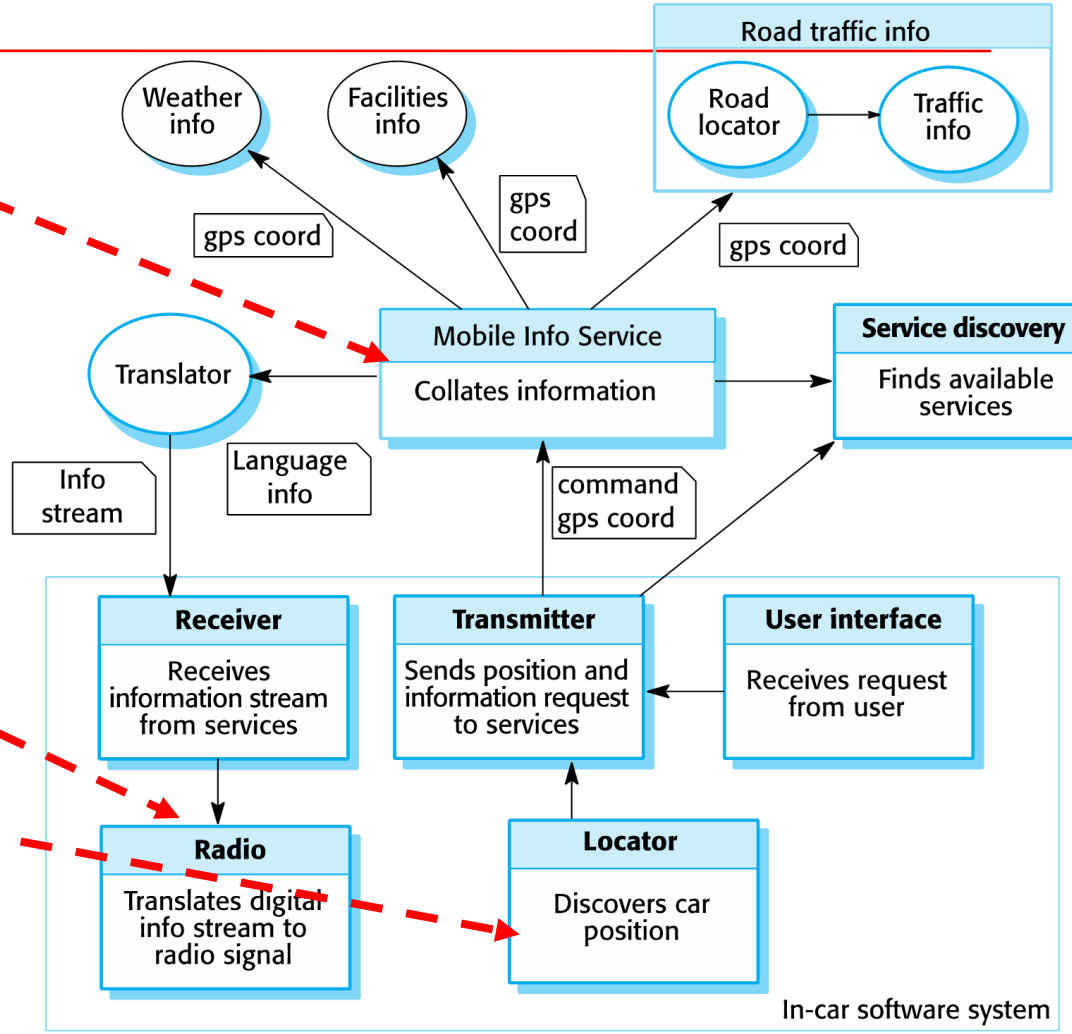
- Păstrare investiție în sisteme legacy.

- Facilitare *calculul inter-organizational* prin simplificarea schimbului de informaţii.

Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

Exemplu : SCENARIU CU UTILIZARE SERVICII

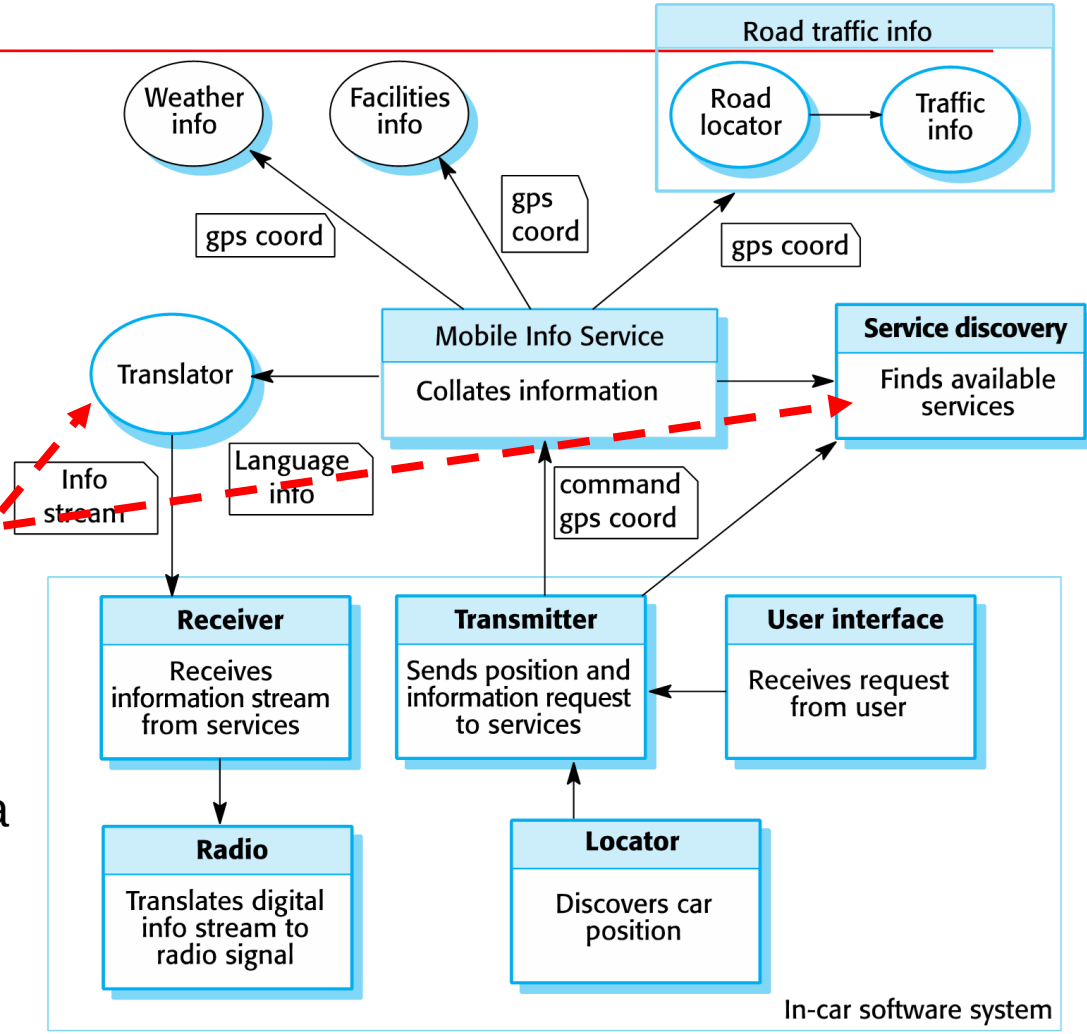
- Un sistem de informare pentru autoturisme oferă informații despre meteo, condiții de trafic și resurse locale.
- Acesta este legat la sistemul audio al autoturismului astfel încât informația este transmisă audio pe un canal specific.
- Autoturismul este echipat cu receptor GPS pentru descoperirea poziției și, pe baza acesteia, sistemul accesează mai multe servicii de informații.
- Informațiile pot fi livrate în limba preferată de șofer.



Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

Exemplu : AVANTAJELE UTILIZĂRII DE SERVICII

- Nu este necesar să se decidă la programarea sau la instalarea aplicației ce furnizor de servicii va fi folosit sau ce servicii anume vor fi utilizate.
- Aplicația folosește un serviciu de descoperire servicii pentru a găsi cel mai potrivit serviciu de informații pentru poziția curentă a autoturismului și se va conecta la acesta.
- Folosește un serviciu de traducere în limba persoanelor ce nu vorbesc limba locului în care autoturismul se află la un moment dat.



Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

INGINERIA SOFTWARE ORIENTATĂ PE SERVICII

- Dezvoltare la fel de semnificativă ca dezvoltarea OO.
- Construirea de aplicații bazată pe servicii permite companiilor și altor organizații să coopereze și să-și folosească reciproc funcții business.
- Aplicațiile bazate pe servicii pot fi construite prin legarea de servicii de la diverși furnizori, folosind fie un limbaj standard de programare fie un limbaj specializat de descriere fluxuri de activități (workflow language).
- Abordările curente în ingineria software evoluează în sensul includerii dezvoltării orientate pe servicii :

Dezvoltare de software pentru reutilizare

– Ingineria serviciilor = dezvoltarea de servicii fiabile, reutilizabile

Dezvoltare de software cu reutilizare

– Dezvoltarea de software folosind servicii = dezvoltarea de software fiabil în care serviciile sunt componentele fundamentale.

Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

ARHITECTURI SOFTWARE ORIENTATE PE SERVICII

SOA (Service Oriented Architecture)

Caracteristici fundamentale:

- Arhitecturi de *sisteme distribuite* în care componentele sunt *servicii de sine stătătoare*.
- Serviciile se pot executa pe diferite *calculatoare* de la diferiţi *furnizori de servicii*.
- Se utilizează *protocoale specializate* pentru comunicarea şi schimbul de informaţii între servicii.

Evaluare formativă

1. Indicați două beneficii ale abordării orientate pe servicii în dezvoltarea aplicațiilor software.

2. Explicați următoarea afirmație :

“Legarea serviciului la aplicație se face la instalare sau la execuție. Rezultă că aplicațiile pot fi reactive și își pot adapta operarea la modificările din contextul de execuție.“

https://forms.gle/jueZLYLs9QZA16jv6

PLAN CURS

Arhitecturi software orientate pe servicii

Servicii web clasice

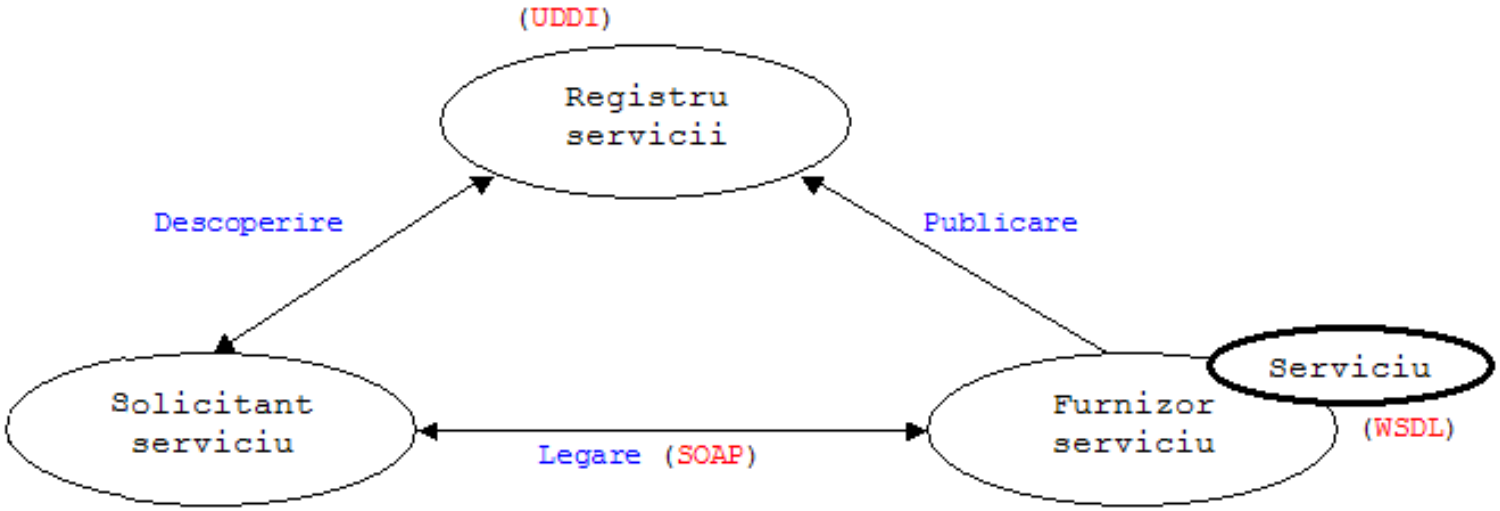
Servicii web RESTful

Ingineria serviciilor

Compunerea serviciilor

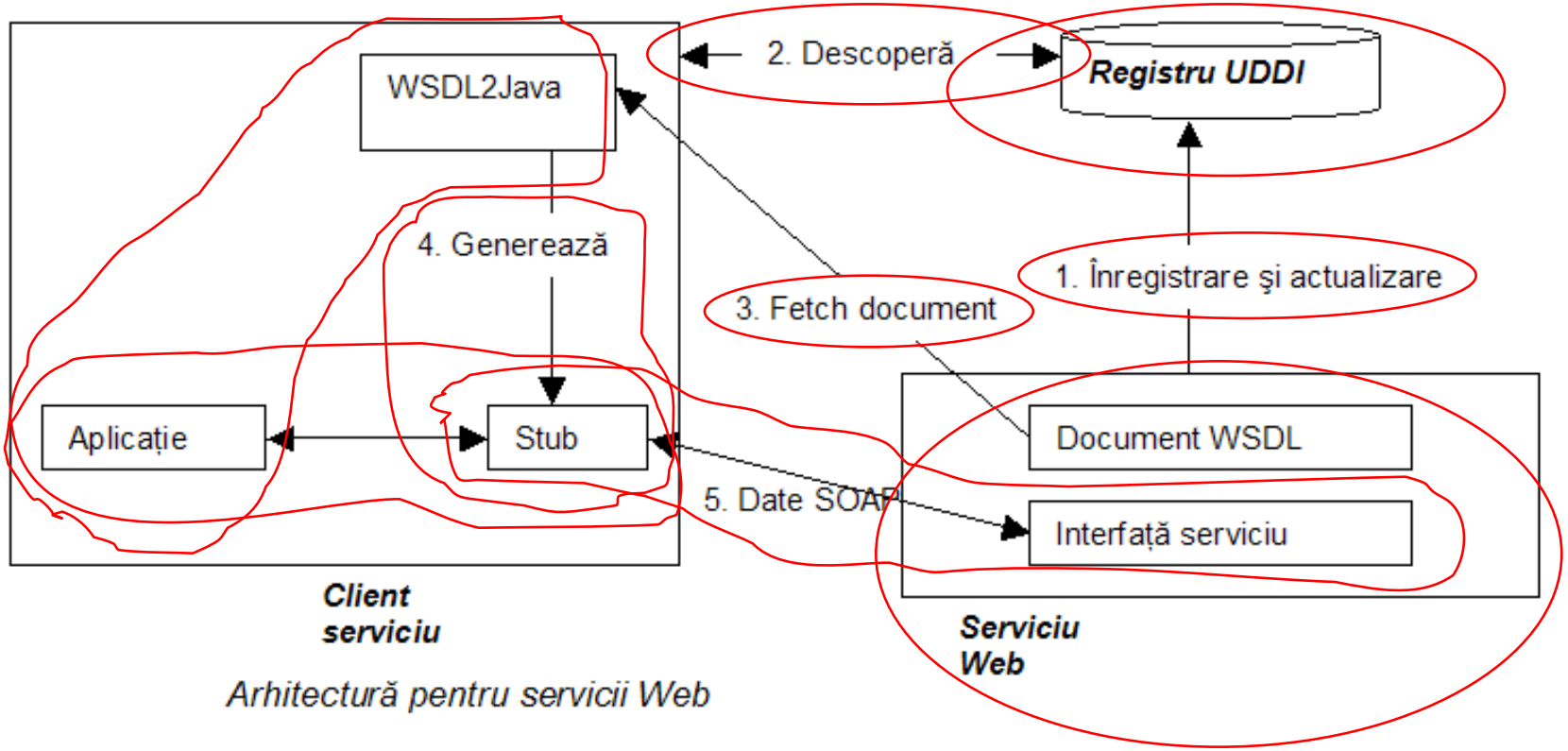
Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

ARHITECTURĂ SOFTWARE GENERICĂ ORIENTATĂ PE SERVICII



Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

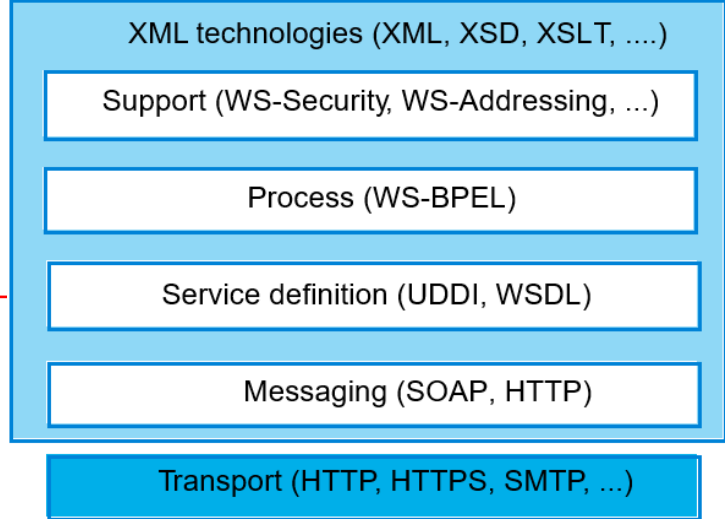
ARHITECTURĂ PENTRU SERVICII WEB



Arhitectură pentru servicii Web

Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

STANDARDE PENTRU SERVICII WEB



SOAP (Simple Object Acess Protocol)

–Standard pentru schimbul de *mesaje*, suport pentru *comunicarea* cu serviciul.

WSDL (Web Service Description Language)

–Standard pentru descrierea *interfeţei* şi *legăturilor* unui serviciu Web.

UDDI (Universal Description, Discovery and Integration)

–Standard pentru definirea componentelor *specificaţiei* serviciului utilizate în *descoperirea* serviciului.

WS-BPEL (Web Service Business Process Execution Language)

–Limbaj standard de descriere a *fluxului de activităţi* (workflow), utilizat la definirea *compunerii* serviciilor.

Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

SERVICII – COMPONENTE REUTILIZABILE

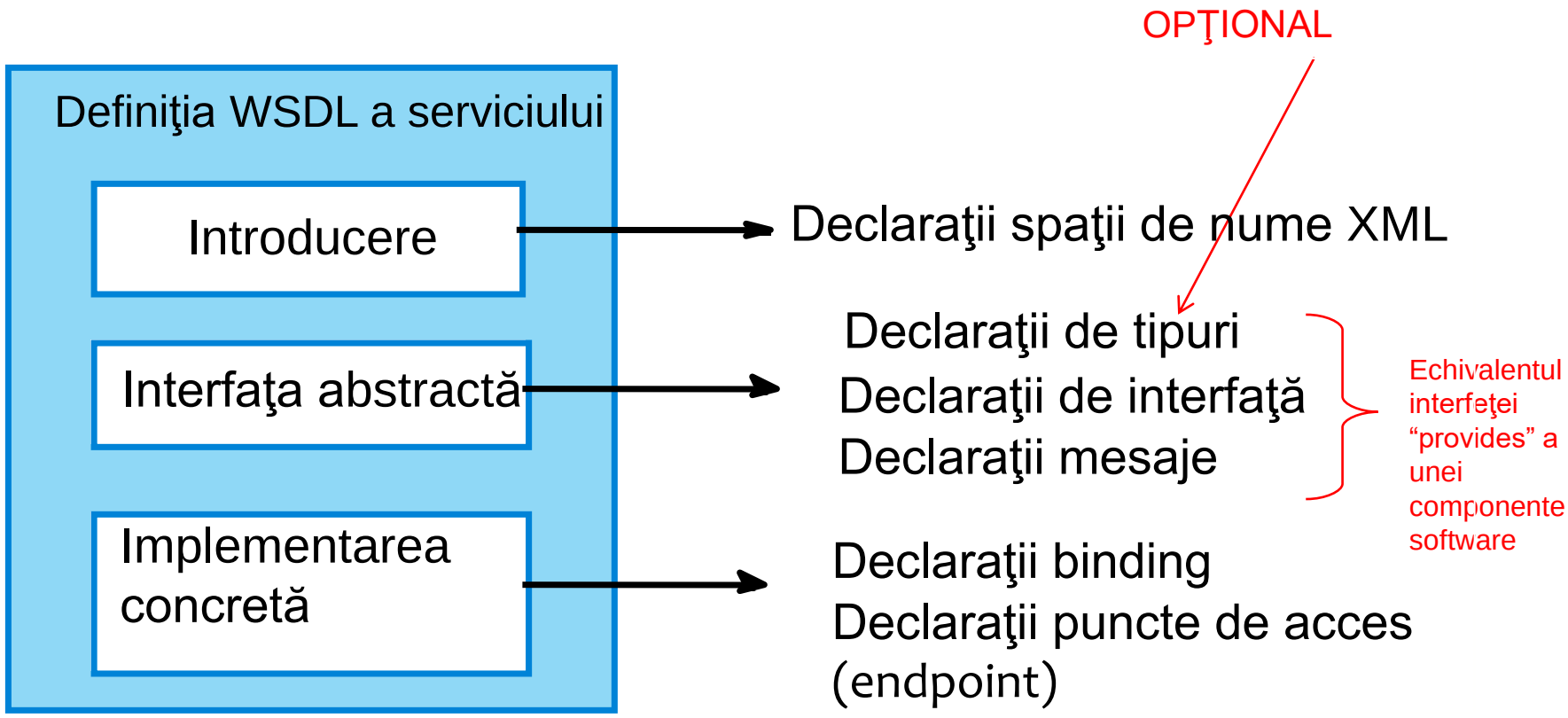
WSDL – Web Service Description Language

Definirea unui serviciu

Elementele definiţiei:

- ***Interfaţa*** : *operaţiile* suportate de serviciu şi *mesajele* trimise şi recepţionate de serviciu.
- ***Modul de accesare*** a serviciului (binding): corespondenţa dintre interfaţa (abstractă) şi un set concret de *protocoale*.
- ***Serviciul*** : localizarea serviciului, exprimată de regulă sub formă de *URI* (Universal Resource Identifier)

ORGANIZAREA SPECIFICAȚIEI WSDL



Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

WSDL – ELEMENTE COMPONENTE

Elementele definiţiei WSDL 2.0:

CE = interfața

- Operațiile
- Formatele mesajelor trimise și recepționate de serviciu

CUM = binding

- Corespondența între interfața abstractă și un set concret de protocoale; specifică detaliile tehnice ale comunicării cu serviciul Web.

UNDE

- Localizarea implementării serviciului (endpoint)

Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

WSDL – ELEMENTE COMPONENTE

Elementele definiţiei WSDL 2.0:

ABSTRACTE — ***CE*** = interfața

- **Types** – container pentru definiţiile de tip
- **Message** – definiţie abstractă de tipuri pentru datele comunicate
- **Operation** – definiţie abstractă a unei acţiuni suportată de serviciu
- **PortType** – set de operaţii (suportate de unul sau mai multe puncte de acces)

CONCRETE — ***CUM*** = binding

- **Binding** – protocol concret şi specificaţie pentru format de date pentru un anume PortType

UNDE **-** puncte de acces

- **Port** – punct de acces definit ca şi combinaţie de binding şi adresă de reţea
- **Service** – colecţie de puncte de acces

Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

WSDL – ELEMENTE ABSTRACTE - exemplu

```
<types>
  <s:element name="Sub">
    <s:complexType>
      <s:sequence>
        <s:element minOccurs="1" maxOccurs="1" name="x" type="s:double" />
        <s:element minOccurs="1" maxOccurs="1" name="y" type="s:double" />
      </s:sequence>
    </s:complexType>
  </s:element>
  <s:element name="SubResponse">
    <s:complexType>
      <s:sequence>
        <s:element minOccurs="1" maxOccurs="1" name="SubResult"
                                                type="s:double" />
      </s:sequence>
    </s:complexType>
  </s:element>
</types>
```

Container pentru definiţiile tipurilor de date

exemplu

```
<?xml version="1.0" encoding="utf-8" ?>
<definitions...
        target Name space="http://OnLineMath.com"
            xmlns="http://schemas.xmlsoap.org/wsdl/">
    <types>
     // definiţii tipuri de date
    </types>

    // definiţii mesaje

    <portType>
     // set abstract de operaţii suportate de serviciu
    </portType>

    <binding>
     // legare portType la o specificaţie concretă pentru protocol şi format de date
    </binding>

    <service>
     // colecţie de porturi; localizare cu URL
    </service>
</definitions>
```

exemplu

exemplu

```
<message name:=:"SubSoapIn">
  <part name:"parameters" element="s0:Sub" />
</message>
<message name="SubSoapOut">
  <part name="parameters" element="s0:SubResponse" />
</message>
```

Definiţie abstractă de tipuri pentru datele comunicate

```
<portType name="ServicelSoap">
  <operation name="Sub">
    <input message="s0:SubSoapIn" />
    <output message="s0:SubSoapOut" />
  </operation>
</portType>
```

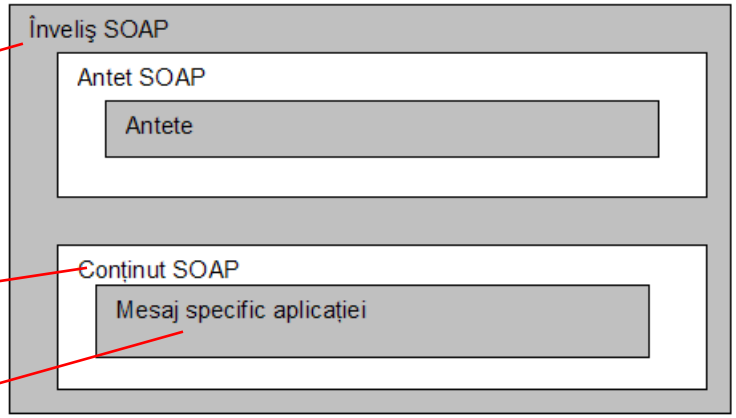
Set de operaţii

Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

SOAP exemplu

Mesaj SOAP pentru lansarea operaţiei **Sub**

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema"
               xmlns: soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <Sub xmlns="http://OnLineMath.com">
        <x> 3 </x>
        <y> 4 </y>
    </Sub>
  </soap:Body>
</soap:Envelope>
```



Structura unui mesaj SOAP

Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

WSDL – ELEMENTE CONCRETE - exemplu

```
<portType name="ServicelSoap">
  <operation name="Sub">
    <input message="s0:SubSoapIn" />
    <output message="s0:SubSoapOut" />
  </operation>
</portType>
```

```
<?xml version="1.0" encoding="utf-8" ?>
<definitions...
        target Name space="http://OnLineMath.com"
            xmlns="http://schemas.xmlsoap.org/wsdl/">
    <types>
     // definiții de tip
    </types>

  // definiții mesaje

    <portType>
     // set abstract de operații suportate de serviciu
    </portType>

    <binding>
     // legare portType la o specificație concretă pentru protocol şi format de date
    </binding>

    <service>
     // colecție de porturi; localizare cu URL
    </service>
</definitions>
```

Protocol concret + specificaţie pentru format de date pentru un anume PortType

exemplu

exemplu

```
<binding name="Service 1 Soap" type="s0:ServicelSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
style="document" />
    <operation name="Sub">
        <soap:operation soapAction="http://OnLineMath.com/Sub"
style="document" />
            <input>
              <soap:body use="literal" />
            </input>
            <output>
              <soap:body use="literal" />
            </output>
    </operation>
  </binding>
```

Service – colecţie de puncte de acces.

```
<service name="Service1">
    <port name="Service 1 Soap" binding="s0:Service 1 Soap">
        <soap:address location="http://OnLineMath.com/WSCalc/Service1.asmx"/>
    </port>
</service>
```

Port – Punct de acces : binding + adresă de reţea.

Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

WSDL – ELEMENTE CONCRETE - exemplu

```
<?xml version="1.0" encoding="utf-8" ?>
<definitions...
        target Name space="http://OnLineMath.com"
            xmlns="http://schemas.xmlsoap.org/wsdl/">
    <types>
     // definiții de tip
    </types>

   // definiții mesaje

    <portType>
     // set abstract de operații suportate de serviciu
    </portType>

    <binding>
     // legare portType la o specificație concretă pentru protocol şi format de date
    </binding>

    <service>
     // colecție de porturi; localizare cu URL
    </service>
</definitions>
```

```
<portType name="ServicelSoap">
  <operation name="Sub">
    <input message="s0:SubSoapIn" />
    <output message="s0:SubSoapOut" />
  </operation>
</portType>
```

```
<binding name="Service 1 Soap" type="s0:ServicelSoap">
   <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
style="document" />
   <operation name="Sub">
       <soap:operation soapAction="http://OnLineMath.com/Sub"
style="document" />
          <input>
            <soap:body use="literal" />
          </input>
          <output>
            <soap:body use="literal" />
          </output>
   </operation>
</binding>
```

```
<service name="Servicel">
  <port name="Service 1 Soap" binding="s0:Service 1 Soap">
    <soap:address location="http://OnLineMath.com/WSCalc/Service1.asmx"/>
  </port>
</service>
```

Descrie:

•Modul de împachetare a mesajelor şi protocolul de comunicare utilizat (implicit SOAP)

•Suplimentar - cum sunt incluse informaţii suport (ex. credenţiale de securitate sau identificatori de tranzacţie)

Localizarea fizică a serviciului exprimată, ca adresă de resursă Internet, prin URI.

Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

PLAN CURS

Arhitecturi software orientate pe servicii

Servicii web clasice

Servicii web RESTful

Ingineria serviciilor

Compunerea serviciilor

Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

SERVICII RESTful

Standardele clasice pentru servicii Web sunt considerate ”heavyweight”, prea generale și ineficiente.

REST (Representational State Transfer) este un stil arhitectural bazat pe transferul de *reprezentare a unei resurse*,* de la un server la un client.

https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

REST = stil arhitectural aflat la baza întregului Web.

*”A resource is a conceptual mapping to a set of entities, not the entity that corresponds to the mapping at any particular point in time.

More precisely, a resource *R* is a temporally varying membership function $M_R(t)$, which for time *t* maps to a set of entities, or values, which are equivalent. The values in the set may be *resource representations* and/or *resource identifiers*. A resource can map to the empty set, which allows references to be made to a concept before any realization of that concept exists -- a notion that was foreign to most hypertext systems prior to the Web [61]. Some resources are static in the sense that, when examined at any time after their creation, they always correspond to the same value set. Others have a high degree of variance in their value over time. The only thing that is required to be static for a resource is the semantics of the mapping, since the semantics is what distinguishes one resource from another. ”

Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

SERVICII RESTful

REST = stil arhitectural aflat la baza întregului Web.

O parte din ideile acestui stil au fost preluate și utilizate în domeniul arhitecturilor orientate pe servicii (SOA) pentru a defini o metodă mai simplă decât SOAP/WSDL pentru implementarea de servicii web.

Your API isn't RESTful — And That's Good | by Trevor Reed | Medium

Serviciile RESTful implică *overhead mai redus* decât serviciile clasice și sunt utilizate de multe organizații care implementează sisteme bazate pe servicii.

SERVICII RESTful
MODELUL CONCEPTUAL

REST (Representational State Transfer) ***adaptat*** la servicii Web.

- Stil arhitectural – bazat pe *client-server* şi protocol de comunicare *stateless* (HTTP)
- *Datele* şi *funcţionalităţile* sunt considerate RESURSE
- Clienţii şi serverele transferă reprezentări ale resurselor folosind *o interfaţă* şi un *protocol standard (HTTP(s)).*

  –***Interfaţa*** : set uniform de *operaţii* (GET[1], PUT[2], POST[2], DELETE) suportate de serviciu şi *mesajele (HTTP)* trimise şi recepţionate de serviciu.

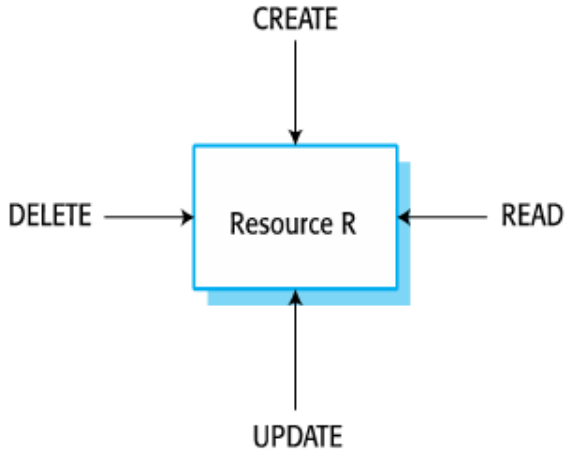
  –***Modul de accesare*** a serviciului (binding): cerere HTTP

  –***Serviciul*** : localizarea serviciului, exprimată sub formă de *URI* (Universal Resource Identifier)

[1] Extrage starea curentă a resursei

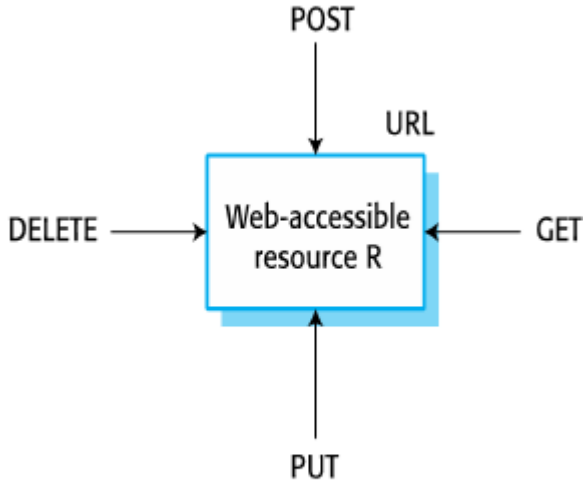
[2] Transferă resursei o nouă stare

OPERAȚII CU RESURSE



(a) General resource actions

Create – creare/activare resursă
Read – returnare reprezentare a unei resurse
Update – modificare valoare resursă
Delete – resursa devine inaccesibilă



(b) Web resources

Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

METODELE HTTP 1.1

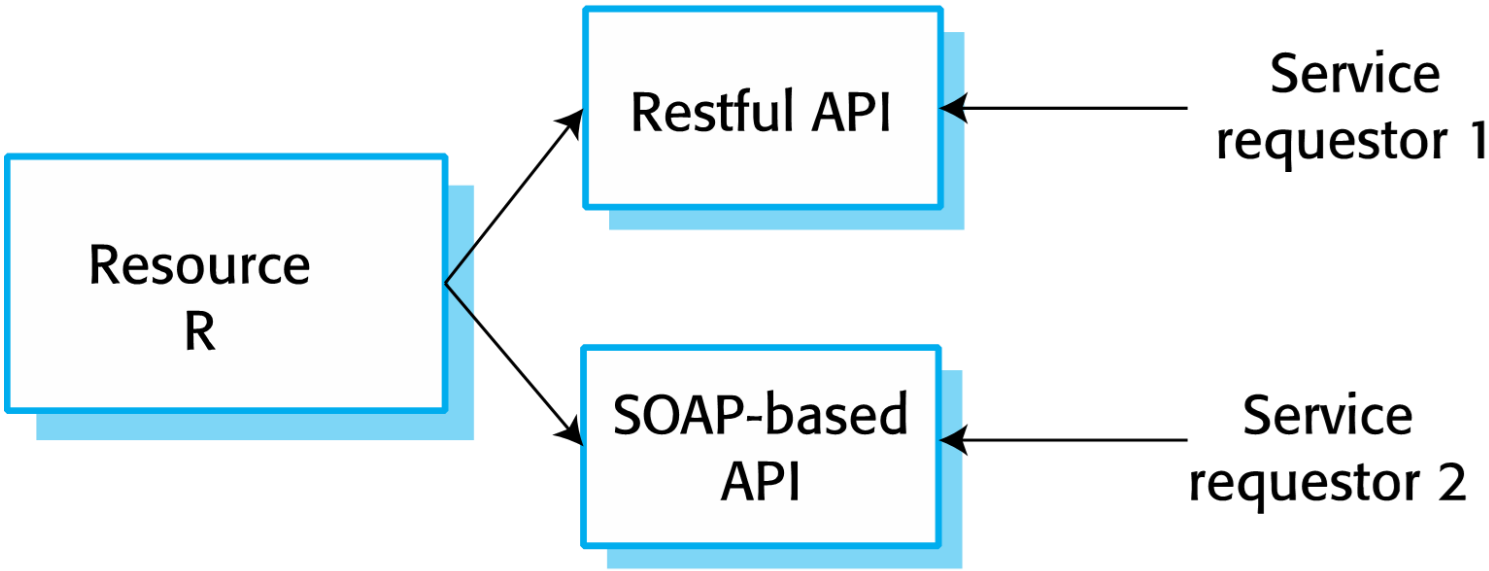
https://tools.ietf.org/html/rfc7231

GET – solicită o reprezentare curentă a resursei
HEAD – solicită metadate despre o reprezentare curentă a resursei
(utilă în testare link-uri)
POST – resursa procesează reprezentarea primită conform semanticilor specifice ale resursei
PUT – creare/modificare stare a resursei; fără efecte colaterale
PATCH – ***descrie*** modificări (parțiale) de aplicat la o resursă; poate avea efecte colaterale
DELETE - ștergere resursă.

TRACE – reflectă cererea recepționată de resursă; se pot observa posibile modificări introduse de servere intermediare
OPTIONS – returnează lista opțiunilor de comunicare cu o resursă sau cu serverul
CONNECT – creare conexiune de tip tunel pentru a facilita comunicare criptată (HTTPS) prin HTTP proxy necriptat

API-uri bazate SOAP și RESTful

Aceeași resursă poate fi expusă prin API RESTful și prin API bazat pe SOAP.



Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

SERVICII RESTful

REST (Representational State Transfer) adaptat la servicii Web.

- Resursele sunt *decuplate* de reprezentări ⇒ conţinutul lor poate fi accesat în diverse formate (ex. HTML, XML, PDF, text, JSON, etc.)
- *Metadate* despre resurse disponibile, folosite pentru control caching, detectare erori transmisie, negociere format reprezentare, autentificare, control drepturi de acces.
- Fiecare interacţiune cu o resursă este *stateless*, dar există soluţii pentru interacţiuni stateful (ex. cookies, transfer stare prin mesajul de răspuns, etc.)
- URI-uri – *intuitive* şi *organizate* în structuri de tip director

–ierarhie cu rădăcină unică

–implementarea unei resurse poate fi modificată fără implicaţii asupra URI

ex.

- definire set structurat http://www.myservice.org/discussion/topics/**{topic}**

- adresare resursă din setul structurat : http://www.myservice.org/discussion/topics/**serviciiWeb**

URI Template
https://tools.ietf.org/html/rfc6570

Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

Evaluare formativă

1. Elementele definiției unui serviciu software sunt interfața (CE), modul de acces (CUM) și punct(e) de acces (UNDE). Explicați ce simplificări aduce abordarea RESTful comparativ cu WSDL ?

CE = interfața

- Operațiile
- Formatele mesajelor trimise și recepționate de serviciu

CUM = binding

- Corespondența între interfața abstractă și un set concret de protocoale; specifică detaliile tehnice ale comunicării cu serviciul Web.

UNDE

- Localizarea implementării serviciului (*endpoint*).

https://forms.gle/XzHhbxNNc5VshFii9

API Description Language (API DL)
Exemple (1)

API DL – limbaj formal (echivalent IDL*, orientat pe servicii RESTful) - descriere structurată, utilizabilă de persoane și mașini.

Exemplu :
RAML – RESTful API Modeling Language (o aplicație a specificației YAML* 1.2)

Oferă mecanisme pentru :

- Definire RESTful API
- Creare *stub*(ciot) de cod sursă client/server
- Documentare completă API pentru clienți

https://raml.org/

https://www.baeldung.com/raml-restful-api-modeling-language-tutorial

https://github.com/raml-org/raml-spec/blob/master/versions/raml-10/raml-10.md/

*Interface Definition Language – folosit pentru definiții de interfețe nenesare comunicărilor la distanță, în siseme distribuite

**YAML - human friendly data serialization for all programming languages; natural superset of JSON with improved human readability and a more complete information model.
https://yaml.org/ ; https://yaml.org/spec/1.2/spec.html

Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

API Description Language (API DL)
Exemple (1)

API DL – limbaj formal (echivalent IDL, orientat pe servicii RESTful) - descriere structurată, utilizabilă de persoane și mașini.

Exemplu :

RAML – RESTful API Modeling Language (o aplicație a specificației YAML* 1.2)

Exemple de ***instrumente*** existente:

Proiectare : API Designer

Proiectare, construire, testare, documentare : API Workbench

Generare JAX-RS skeleton sau RAML specification : RAML for JAX-RS

Generare documentație : RAML to HTML, raml2pdf

Testare : SoapUI RAML Plugin RAML plugin pentru SoapUI functional API testing suite

Generare cazuri de testare : Vigia

Validare fișiere RAML : RAML Cop

Editare : RAML Sublime Plugin – plugin (syntax highlighter) pentru editorul de texte Sublime

API Description Language (API DL)
Exemple (2)

API DL – limbaj formal (echivalent IDL, orientat pe servicii RESTful) - descriere structurată, utilizabilă de persoane și mașini

Exemplu :

OpenAPI –IDL pentru apel servicii oferite prin REST APIs

Utilizat de instrumente :

- Generare documentație API pentru clienți
- Generare stub(ciot) de cod sursă pentru client și server în diferite limbaje
- Testare, etc

https://www.openapis.org/

http://spec.openapis.org/oas/v3.0.2

PLAN CURS

Arhitecturi software orientate pe servicii

Servicii web clasice

Servicii web RESTful

Ingineria serviciilor

Compunerea serviciilor

INGINERIA SOFTWARE-lui ORIENTAT PE SERVICII - DOMENII

- Dezvoltare de servicii reutilizabile (ingineria serviciilor)
- Dezvoltare de software în care serviciile sunt componente fundamentale.

Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

INGINERIA SERVICIILOR

Ingineria serviciilor = procesul de ***dezvoltare*** de servicii,

pentru ***reutilizarea*** acestora în aplicaţii orientate pe servicii.

- Serviciul trebuie proiectat sub formă de ***abstractizare reutilizabilă*** ce poate fi utilizată în diferite sisteme.
- Atribute suplimentare dezirabile: ***robusteţe, fiabilitate***.
- Serviciul trebuie ***documentat*** pentru a fi descoperit şi înţeles de potenţialii utilizatori.

Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

INGINERIA SERVICIILOR

Ingineria serviciilor = procesul de *dezvoltare* de servicii,

pentru *reutilizarea* acestora în aplicaţii orientate pe servicii.

Etapele procesului

1. Identificarea serviciului candidat

- Identificare serviciu posibil și definirea cerințelor pentru acesta

2. Proiectarea serviciului

- Proiectarea interfeței logice a serviciului
- Proiectarea interfețelor de implementare a serviciului (SOAP și/sau RESTful)

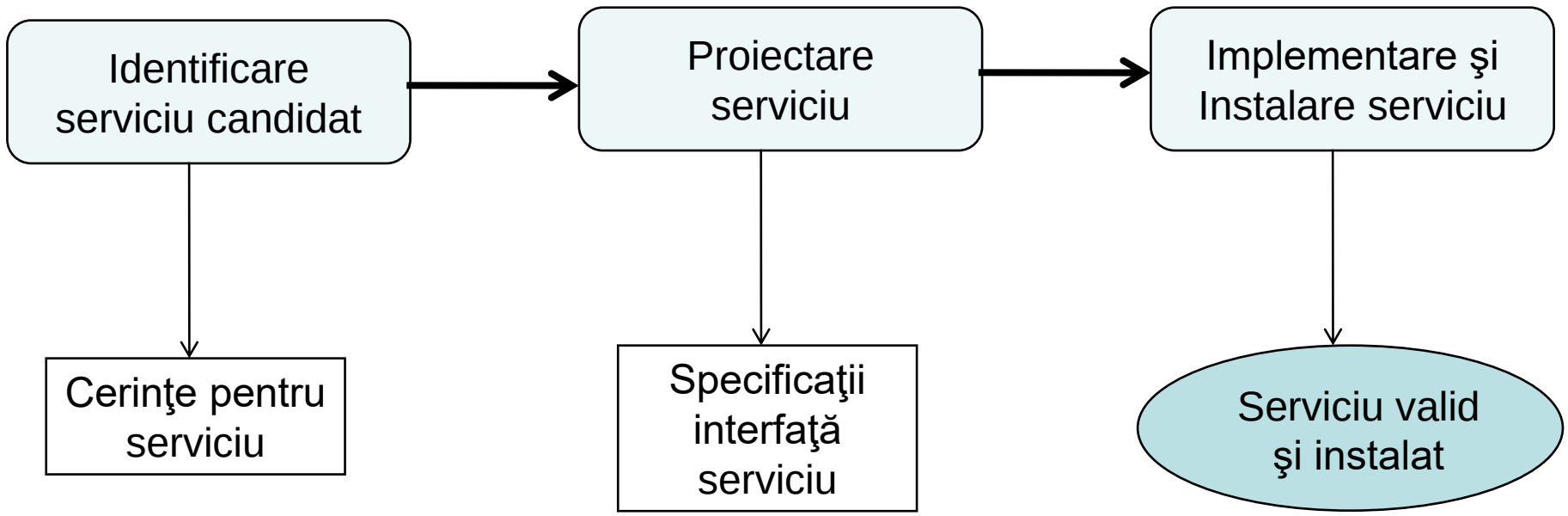
3. Implementarea serviciului

- Implementare și testare serviciu
- Instalare serviciu

Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

INGINERIA SERVICIILOR

PROCESUL INGINERIEI SERVICIILOR



Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

INGINERIA SERVICIILOR

IDENTIFICAREA SERVICIULUI CANDIDAT

Înțelegerea procesului business al organizației și identificarea serviciilor reutilizabile ce pot sprijini acest proces.

Tipuri fundamentale de servicii:

- **Utilitar** – implementează funcţionalitate generală folosită de diferite procese business (ex. convertor valutar).
- **Business** – asociat unei anumite funcţii business (ex. înregistrarea studentului într-o universitate).
- **De coordonare (de proces)** – suport pentru procesele compuse (ex. prelucrare ordin: plasare ordin, acceptare produse, realizare plată).

INGINERIA SERVICIILOR

Clasificare ortogonală (cu precedenta):

- Serviciu *orientat task* – asociat unei activităţi.
- Serviciu *orientat entitate* – asociat unei entităţi business.

Exemple	Utilitar	Business	De coordonare (proces)
Task	-Convertor valutar -Localizator angajat	-Validare formular creanțe -Verificare rating credit	-Procesare plată creanțe -Plată furnizor extern
Entitate	-Verificare stil document -Convertire formular Web în XML	-Formular de plată -Formular aplicație student	

–Obs. Serviciile de coordonare sunt întotdeauna orientate task

Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

INGINERIA SERVICIILOR

Obiectiv: identificare servicii

- coerente logic
- independente
- reutilizabile

Metodă principală:

- Identificarea entităţilor şi a activităţilor business.
- Extragerea posibililor candidaţi.
- Analizarea candidaţilor în vederea selecţiei.

Identificarea serviciilor se bazează pe abilităţi şi pe experienţă.

Ieşirea procesului de selecţie:

- setul serviciilor identificate
- cerinţele (funcţionale şi extra-funcţionale) asociate serviciilor din set

Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

INGINERIA SERVICIILOR

Întrebări posibile pentru analizarea candidaţilor:

1.Pentru servicii entitate: este acesta asociat cu *o singură entitate logică* utilizată în *diferite procese* business ? Care sunt *operaţiile* realizate asupra acelei entităţi ?

2.Pentru servicii task: este *aceeaşi activitate* realizată de *persoane(roluri) diferite* din organizaţie ? Sunt aceste persoane dispuse să suporte *standardizarea* acestei activităţi ?

3.Este serviciul *independent* de disponibilitatea altor servicii ?

4.Este necesară *păstrarea stării* serviciului ? Se va folosi o bază de date pentru aceasta ?
(Serviciile care gestionează stare intern sunt mai puţin reutilizabile decât cele la care starea poate fi păstrată extern)

5.Serviciul poate fi utilizat de *clienţi din afara* organizaţiei ?

6.E posibil ca utilizatori diferiţi să aibă *cerinţe extra-funcţionale* pentru serviciu *diferite* ? (în acest caz se vor implementa probabil mai multe versiuni ale serviciului).

INGINERIA SERVICIILOR

IDENTIFICAREA SERVICIULUI CANDIDAT - EXEMPLU

O firmă care vinde echipamente pentru calculatoare are prețuri speciale pentru anumite configurații și anumiți clienți.

Pentru a facilita comandarea automată compania vrea să producă un serviciu catalog care va permite clienților să selecteze echipamentul dorit.

Spre deosebire de un catalog clasic, ordinele nu sunt plasate direct printr-o interfață catalog. Bunurile sunt comandate prin sistemul web de aprovizionare al fiecărei companii care accesează catalogul ca pe un serviciu web.

Majoritatea companiilor au proceduri proprii de bugetare și de aprobare a ordinelor și trebuie aplicat procesul propriu fiecărei companii atunci când se plasează un ordin.

INGINERIA SERVICIILOR

IDENTIFICAREA SERVICIULUI CANDIDAT - EXEMPLU

Serviciu catalog de produse

Cerinţe serviciu - FUNCŢIONALE

-Versiune specifică pentru fiecare client

-Catalogul să poată fi descărcat pentru consultare off-line

-Compararea specificaţiilor şi preţurilor pentru max. 6 produse

-Facilităţi de explorare şi căutare

-Predicţia datei de livrare pentru anumite produse

-Posibilitatea de a plasa comenzi virtuale, ce vor reţine produsele max. 48 de ore şi care vor trebui confirmate în limita celor 48 de ore.

INGINERIA SERVICIILOR

IDENTIFICAREA SERVICIULUI CANDIDAT - EXEMPLU

Serviciu catalog de produse

Cerinţe serviciu – EXTRA-FUNCŢIONALE

-Acces permis doar angajaţilor organizaţiilor acreditate

-Preţuri şi configuraţii confidenţiale pentru fiecare client

-Disponibilitate continuă între orele 7:00 – 11:00.

-Încărcarea maximă de procesare - 10 cereri per secundă.

INGINERIA SERVICIILOR

IDENTIFICAREA SERVICIULUI CANDIDAT – EXEMPLU

Descrierea funcţională a *operaţiilor* serviciului catalog de produse.

Operație	Descriere
CreareCatalog	Creare unei versiuni a catalogului pentru un anumit client. Include un parametru opțional, de creare a unei versiuni PDF ce poate fi descărcată la client.
Comparare	Compararea a max. 4 caracteristici (ex. preț, dimensiune) pentru max. 6 produse
Inspectare	Afișarea tuturor datelor asociate unui anumit produs
Căutare	Căutare în catalog pe baza unei expresii logice preluate la intrare. Afișarea tuturor produselor găsite.
VerificareLivrare	Returnează data preconizată pentru livrarea unui anumit produs, dacă este comandat azi.
CreareOrdinVirtual	Rezervarea produselor ce vor fi comandate de client și oferirea de informații pentru sistemul de aprovizionare al acestuia.

INGINERIA SERVICIILOR

PROIECTAREA INTERFEŢEI SERVICIULUI

- Specificarea *operaţiilor* asociate serviciului şi *mesajelor* schimbate de acesta.
- *Minimizarea* numărului de *mesaje* necesare completării unei cereri de serviciu.
- Posibila includere în mesaje a *informaţiilor de stare* a serviciului, pentru a transfera clientului responsabilitatea gestionării acesteia.

INGINERIA SERVICIILOR

Etapele procesului de proiectare a interfeţei serviciului:

-**Proiectarea logicii interfeţei** :

- operaţii, intrări şi ieşiri, excepţii, plecând de la cerinţele serviciului.

Obs. În proiectarea orientată pe componente se recomandă transferul responsabilităţii tratării excepţiilor către utilizatorul componentei.

-**Proiectarea mesajelor** (abordare SOAP):

- structura şi organizarea mesajelor de intrare şi de ieşire.

Recomandare : reprezentarea mesajelor ca obiecte (UML, limbaj de programare).

- transformarea specificației logice în descriere WSDL

Recomandare : utilizare de instrumente specializate care automatizează această operaţie.

-**Poiectare interfață** (abordare RESTful) :

- corespondența între operațiile serviciului și operații REST (metode HTTP).
- resursele necesare

Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

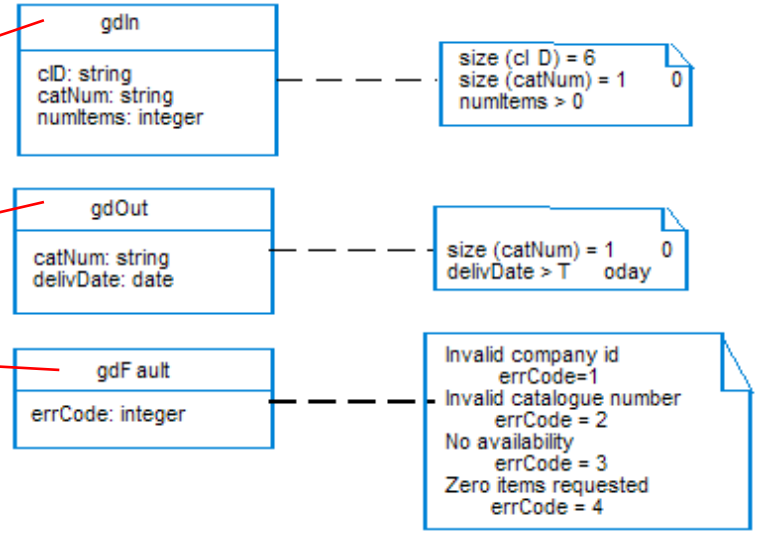
INGINERIA SERVICIILOR

PROIECTAREA INTERFEŢEI SERVICIULUI - EXEMPLU

Proiectarea logicii interfeţei

Operation	Inputs	Outputs	Exceptions
MakeCatalogue	*mcIn* Company id PDF -flag	*mcOut* URL of the catalogue for that company	*mcFault* Invalid company id
Compare	*compIn* Company id Entry attribute (up to 6) Catalogue number (up to 4)	*compOut* URL of page showing comparison table	*compFault* Invalid company id Invalid catalogue number Unknown attribute
Lookup	*lookIn* Company id Catalogue number	*lookOut* URL of page with the item information	*lookFault* Invalid company id Invalid catalogue number
Search	*searchIn* Company id Search string	*searchOut* URL of web page with search results	*searchFault* Invalid company id Badly-formed search string
CheckDelivery	*gdIn* Company id Catalogue number Number of items required	*gdOut* Catalogue number Expected delivery date	*gdFault* Invalid company id Invalid catalogue number No availability Zero items requested
PlaceOrder	*poIn* Company id Number of items required Catalogue number	*poOut* Catalogue number Number of items required Predicted delivery date Unit price estimate Total price estimate	*poFault* Invalid company id Invalid catalogue number Zero items requested

Proiectarea mesajelor (ca ***obiecte***)



Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

INGINERIA SERVICIILOR

INTERFAȚA RESTful : RESURSE

- Pentru fiecare companie există o resursă care reprezintă un **catalog specific**.
- Aceasta are URL-ul de forma

  <base catalog>/<company name>

și va fi creată folosind o operație POST.

- Fiecare element din catalog va avea URL-ul propriu de forma

  <base catalog>/<company name>/<item identifier>

- Operațiile *CheckDelivery și MakeVirtualOrder* necesită o resursă adițională, reprezentând un **ordin virtual.**

INGINERIA SERVICIILOR

INTERFAȚA RESTful: OPERAȚII

Lookup – implementată folosind URL-ul unui element din catalog ca parametru pentru GET.

Search – GET cu URL-ul catalogului companiei și șirul de căutare ca parametru pentru interogare. Se va returna o listă de URL-uri ale elementelor găsite.

Compare – secvență de operații GET pentru a extrage elemente individuale, urmată de operația POST pentru a crea tabela de comparare și un GET final pentru a o returna utilizatorului.

MakeVirtualOrder și *CheckDelivery* necesită o resursă adițională, reprezentând un ordin virtual.

- POST – creare resursă ordin virtual cu numărul necesar de elemente; completate automat ID-ul companiei și data de furnizare calculată.
- GET – extragere reprezentare resursă.

INGINERIA SERVICIILOR

TEMĂ

Creați definiția interfeței serviciului RESTful “Catalog de Produse” din acest exemplu folosind :

1. RAML
2. OpenAPI

INGINERIA SERVICIILOR

IMPLEMENTAREA ŞI INSTALAREA SERVICIILOR

- Programarea serviciilor utilizând un limbaj de programare standard și/sau un limbaj *workflow* (pentru servicii compuse).
- Testarea serviciilor prin crearea de mesaje de intrare şi verificarea corectitudinii mesajelor de ieşire corespunzătoare.
- *Deployment* implică instalarea sa pe un server (Web) şi publicarea serviciului.

DESCRIEREA SERVICIULUI

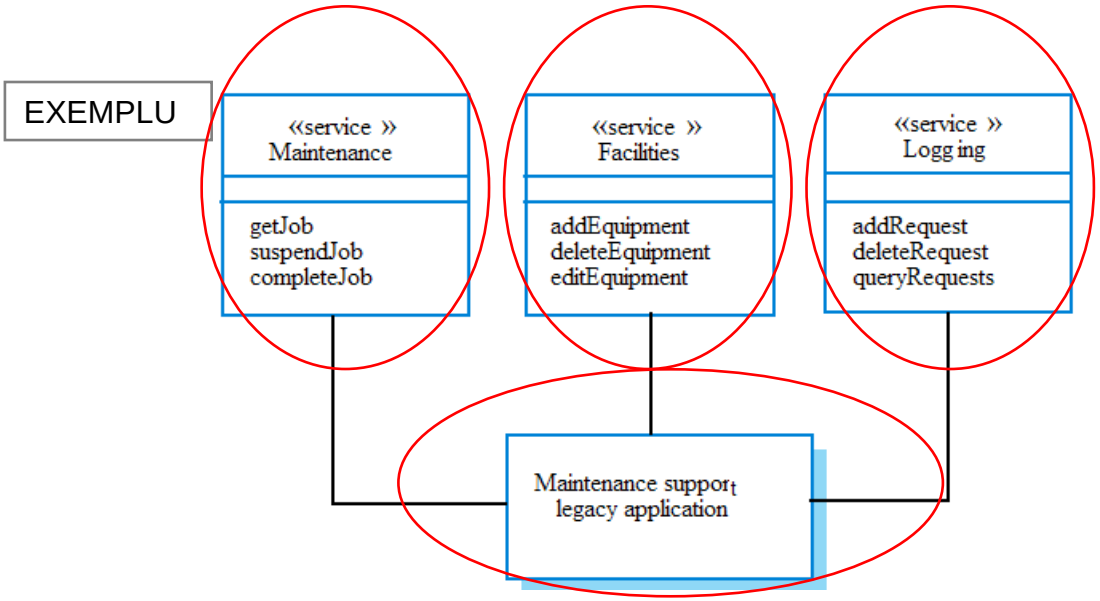
- Detalii despre furnizorul serviciului.
- Descriere informală a funcţionalităţii oferită de serviciu.
- Localizarea specificaţiei WSDL a serviciului, sau informații de accesare pentru servicii RESTful.
- Informaţii despre posibilitatea de abonare la actualizări ale serviciului.

INGINERIA SERVICIILOR

INTEGRAREA SISTEMELOR LEGACY

- O aplicaţie importantă a serviciilor – oferă acces la funcţionalitatea inclusă în sisteme legacy.
- Sistemele legacy oferă funcţionalitate extensivă, ceea ce poate reduce costul implementării serviciului.
- Aplicaţiile externe pot accesa această funcţionalitate prin interfeţele serviciului.

EXEMPLU



Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

PLAN CURS

Arhitecturi software orientate pe servicii

Servicii web clasice

Servicii web RESTful

Ingineria serviciilor

Compunerea serviciilor

Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

DEZVOLTARE DE SOFTWARE ORIENTAT PE SERVICII

Dezvoltare de software orientat pe servicii = *configurarea* şi *compunerea* serviciilor existente pentru a crea noi *servicii compuse* şi *aplicaţii*.

Categorii de servicii :

-servicii dezvoltate special pentru aplicaţie

-servicii business dezvoltate anterior în companie

-servicii de la un furnizor extern.

Compunerea serviciilor se bazează, în general, pe *flux de lucru* (workflow).

–*Workflow* – secvenţă logică de activităţi care, împreună, modelează un proces business coerent.

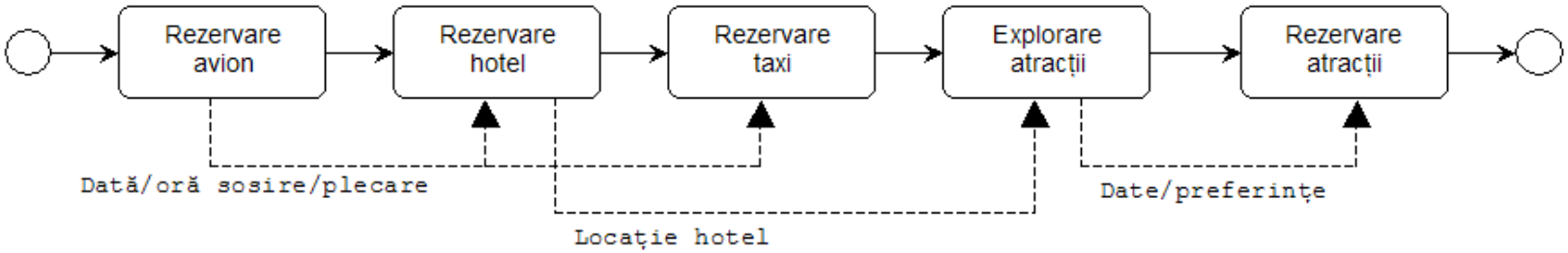
Obs. Un workflow poate include atât servicii SOAP cât și servicii RESTful.

DEZVOLTARE DE SOFTWARE ORIENTAT PE SERVICII

Exemplu: serviciu pentru rezervări, ce permite coordonarea mai multor servicii pentru organizarea unei vacanţe.

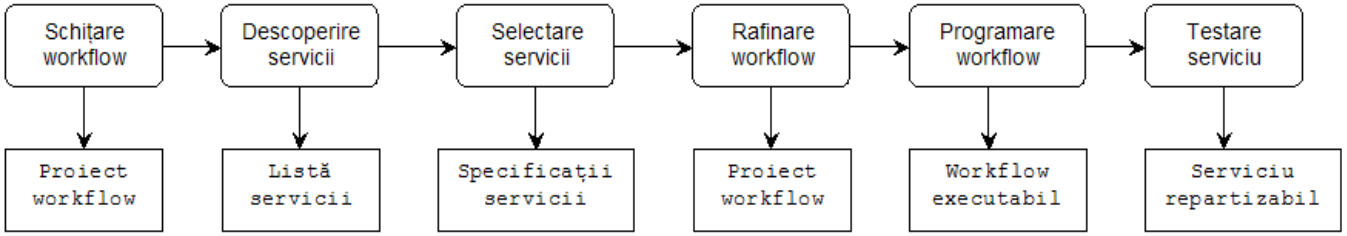
Este necesară *compunerea* (workflow) mai multor servicii oferite de diferiţi furnizori.

Rezultă un *serviciu integrat*.



Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

PROCESUL DE COMPUNERE SERVICII



Schițare workflow – plecând de la cerințele pentru serviciul compus se crează un model ideal al acestuia.

Descoperire servicii – căutare în regiștrii sau cataloage de servicii a serviciilor necesare și extragerea detaliilor de obținere a acestora.

Selectare servicii posibile – pe baza funcționalităților serviciilor descoperite, dar ținând cont de cost și de calitățile serviciului (capacitate de reacție, disponibilitate,etc.)

Rafinarea fluxului de activități (workflow) – detalierea descrierii abstracte și posibil *adăugare/eliminare activități.*

Creare program workflow – transformare flux abstract în program executabil și definirea interfeței serviciului compus.

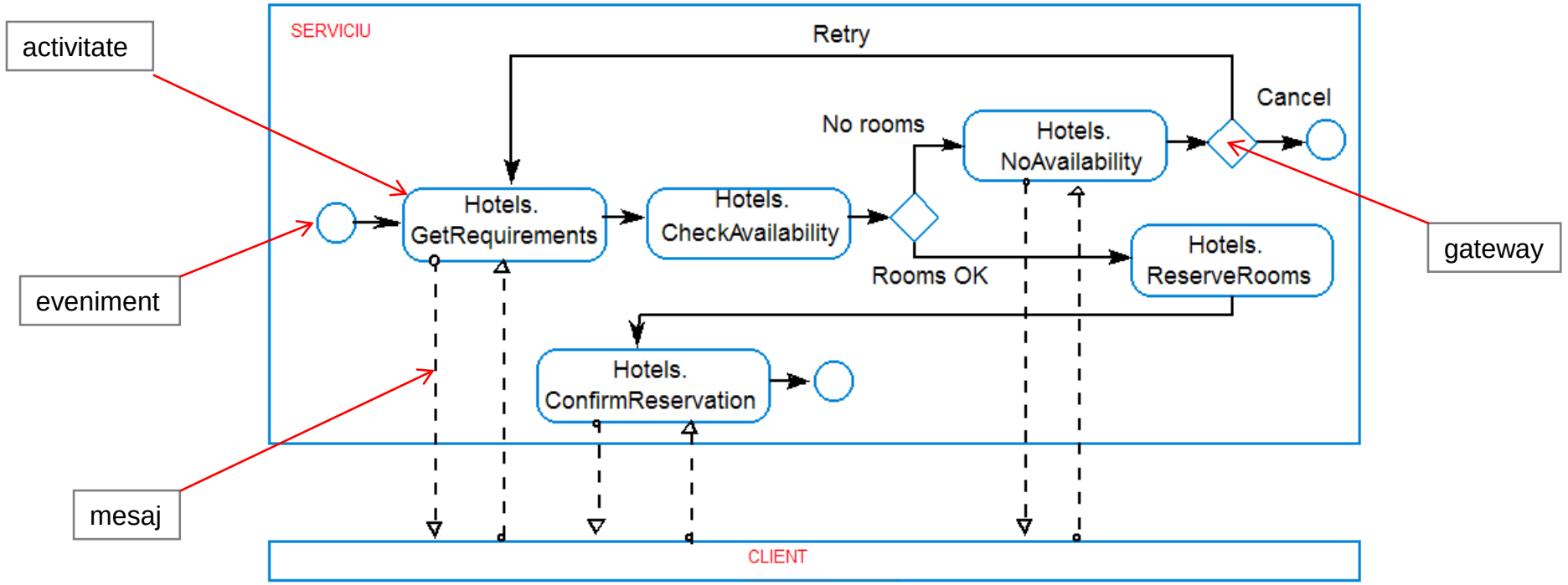
Testarea serviciului compus / aplicației – activitate mult mai complexă dacă se folosesc servicii externe.

Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

DEZVOLTARE DE SOFTWARE ORIENTAT PE SERVICII

DEZVOLTARE DE SOFTWARE ORIENTAT PE SERVICII - EXEMPLU

Workflow pentru rezervare hotel.



Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

DEZVOLTARE DE SOFTWARE ORIENTAT PE SERVICII

PROIECTARE ŞI IMPLEMENTARE WORKFLOW

- WS-BPEL – standard XML pentru specificare workflow

Obs. Descrierile sunt lungi şi greu de citit.

- BPMN (Business Process Modeling Notation) – notaţie grafică

Există instrumente de generare WS-BPEL din BPMN

PROIECTARE ŞI IMPLEMENTARE WORKFLOW

Un workflow poate reprezenta:

- Proces intra-organizaţional – exemplul precedent

- Proces inter-organizaţional –

–creare de fluxuri de activităţi separate pentru fiecare organizaţie,

–legarea lor prin schimb de mesaje.

PROIECTARE ŞI IMPLEMENTARE WORKFLOW

Proces inter-organizational - exemplu

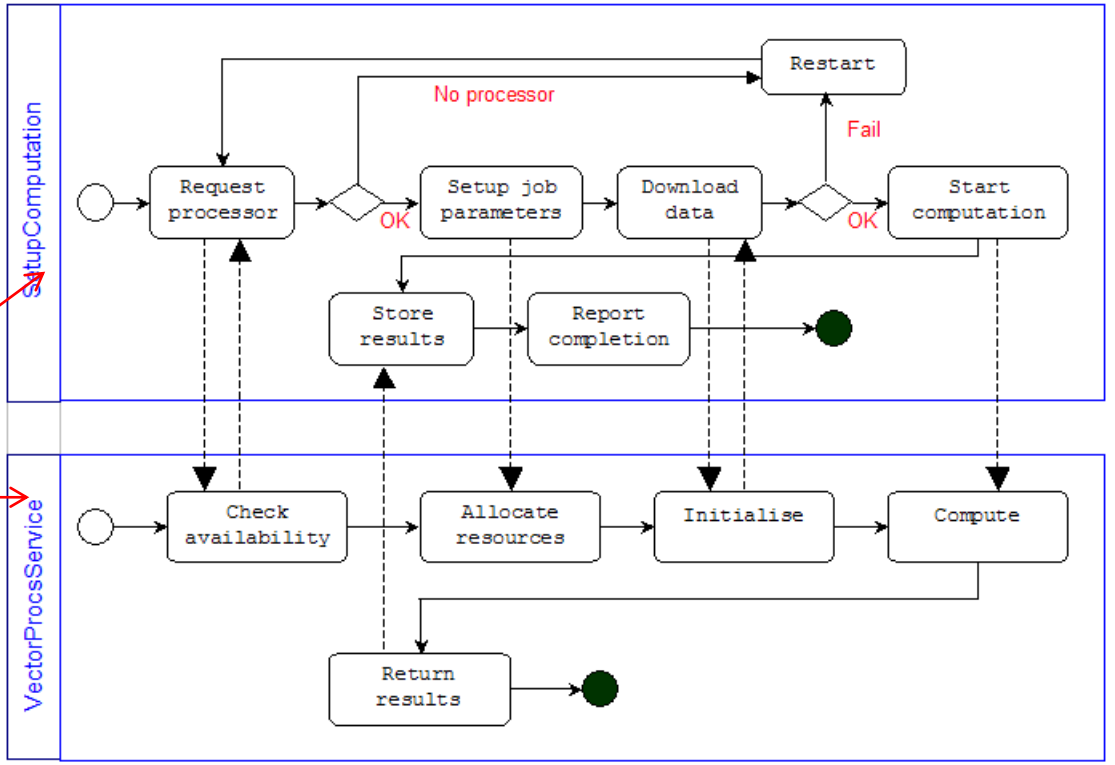
Domeniul: calcul pe grid.

Scop: partajare resurse

Abordare: orientată pe servicii.

Serviciu intermediar: setare calcul.

Serviciu grid: procesor vectorial de înaltă performanţă.



Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

DEZVOLTARE DE SOFTWARE ORIENTAT PE SERVICII

VERIFICARE ŞI VALIDARE SOFTWARE ORIENTAT PE SERVICII

Obiective:

- Găsirea defectelor (verificare).
- Demonstrare că sistemul îndeplineşte cerinţele funcţionale şi extra-funcţionale (validare).

Specific:

- Dificultate datorată faptului că serviciile externe sunt ‘black-boxes’.
- Nu pot fi utilizate tehnicile ce se bazează pe codul sursă.

DEZVOLTARE DE SOFTWARE ORIENTAT PE SERVICII

V&V SOFTWARE ORIENTAT PE SERVICII – PROBLEME

- Serviciile externe pot fi modificate de furnizori, ceea ce invalidează testele realizate deja.
- Legarea dinamică a serviciilor permite utilizarea de servicii variabile; un test realizat într-un anume moment nu este relevant pentru utilizarea ulterioară a serviciilor. ?
- Comportamentul extra-funcţional al unui serviciu este impredictibil deoarece depinde de încărcare.
- Dacă utilizarea unui serviciu se face contra cost, cresc costurile de testare.
- Este dificil de invocat acţiuni compensatorii în servicii externe, deoarece acestea se pot baza pe căderea altor servicii, care nu poate fi simulată.

CONCLUZII

•Ingineria software orientată pe servicii se bazează pe noţiunea că aplicațiile software se pot construi prin *compunere de servicii independente* care *încapsulează funcţionalitate reutilizabilă*.

•Serviciile pot fi implementate în cadrul unei arhitecturi SOA folosind un set de standarde bazate pe XML pentru servicii web: Acestea includ standarde pentru comunicare cu serviciile, pentru definire interfețe, pentru utilizare servicii în fluxuri de activități.

•Alternativ, se poate folosi o arhitectură *RESTful* care se bazează pe resurse și operații standard cu acestea. Abordarea RESTful folosește un *set standard de operații* (POST, GET, PUT, DELETE) și un *protocolul standard* HTTP(S).

•Serviciile se pot clasifica în *utilitare*, *business* şi de *coordonare*.

Adapted after Ian Sommerville, Software Engineering ed 10, cap 18

CONCLUZII

- Procesul ingineriei serviciilor implică *identificarea* serviciilor candidat la implementare, urmată de *definirea interfeţei*, *implementarea*, *testarea* şi *instalarea* fiecărui serviciu.

- Pentru reutilizarea sistemelor software *legacy* în alte aplicaţii, se pot defini la acestea interfeţe de tip serviciu.

- Dezvoltarea de software orientat pe servicii implică crearea de aplicații prin *configurarea* şi *compunerea serviciilor* pentru a crea noi servicii compuse.

- Modelele de procese business definesc activităţile şi schimbul de informaţii în procesele business. Activităţile unui proces business pot fi implementate ca servicii, caz în care modelul unui proces business reprezintă o compunere de servicii.

- Procesul business, incluzând serviciile utilizate în acesta, se poate descrie ca flux de activități folosind limbaje grafice (ex. BPMN).

- Testarea aplicaţiilor orientate pe servicii are probleme specifice ce necesită mai mult decât utilizarea tehnicilor clasice de testare.

Adapted after Ian Sommerville, Software Engineering ed 10, cap 18