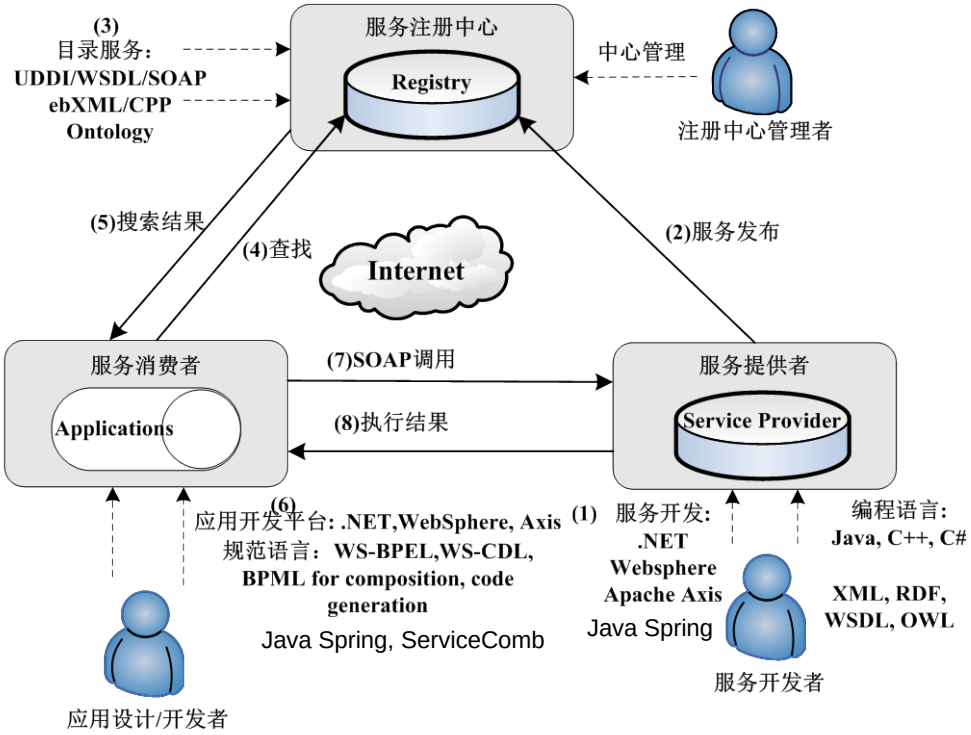
服务计算
Service Computing

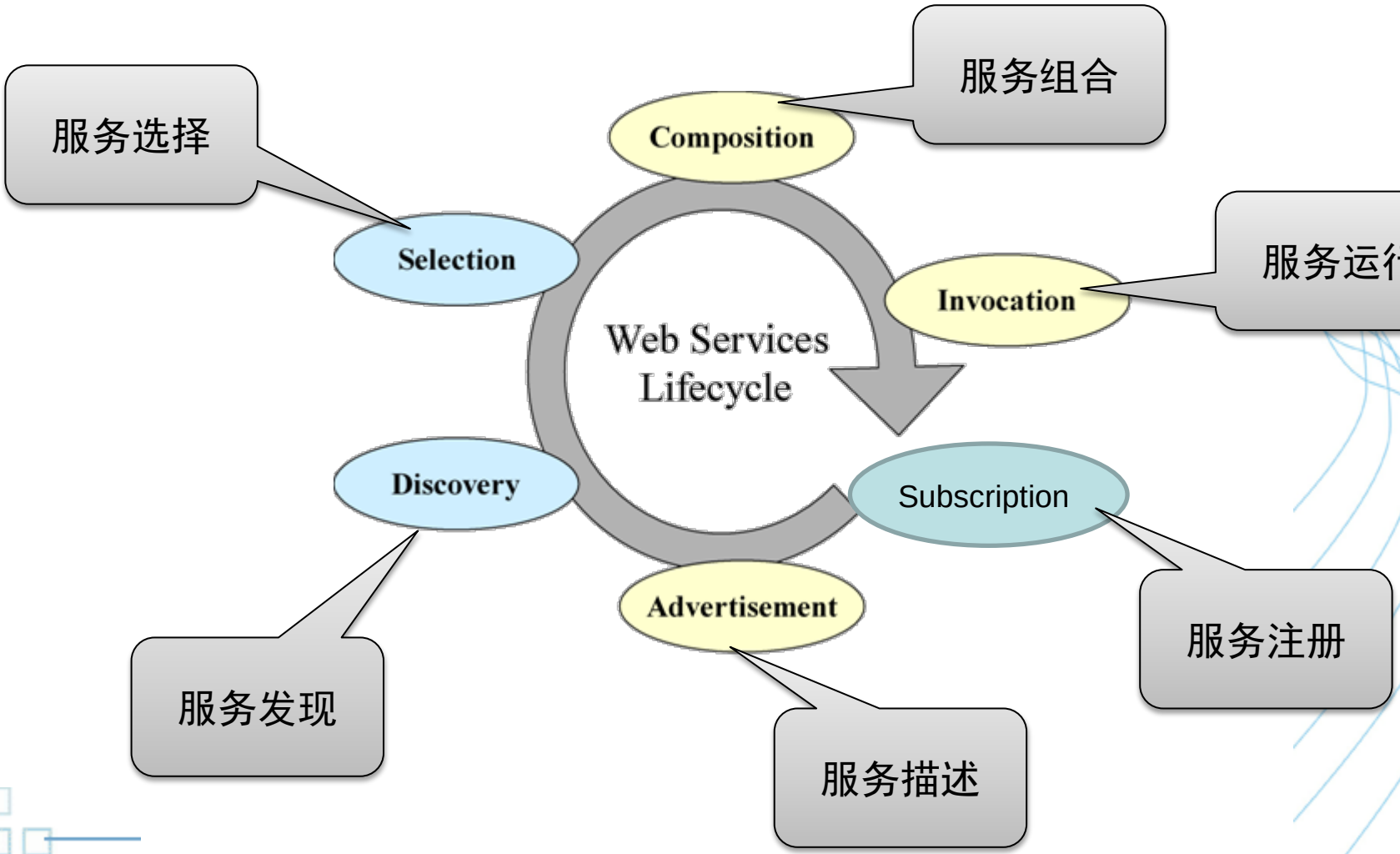
王旭
wangxu@buaa.edu.cn

四、服务注册、描述和发现

服务计算典型模式


(3)
目录服务：
UDDI/WSDL/SOAP
ebXML/CPP
Ontology
服务注册中心
Registry
中心管理
注册中心管理者
(5)搜索结果
(4)查找
Internet
(2)服务发布
服务消费者
Applications
(7)SOAP调用
(8)执行结果
服务提供者
Service Provider
(6)
应用开发平台: .NET,WebSphere, Axis
规范语言：WS-BPEL,WS-CDL,
BPML for composition, code
generation
Java Spring, ServiceComb
(1) 服务开发:
.NET
Websphere
Apache Axis
Java Spring
编程语言:
Java, C++, C#
XML, RDF,
WSDL, OWL
应用设计/开发者
服务开发者

服务Lifecycle



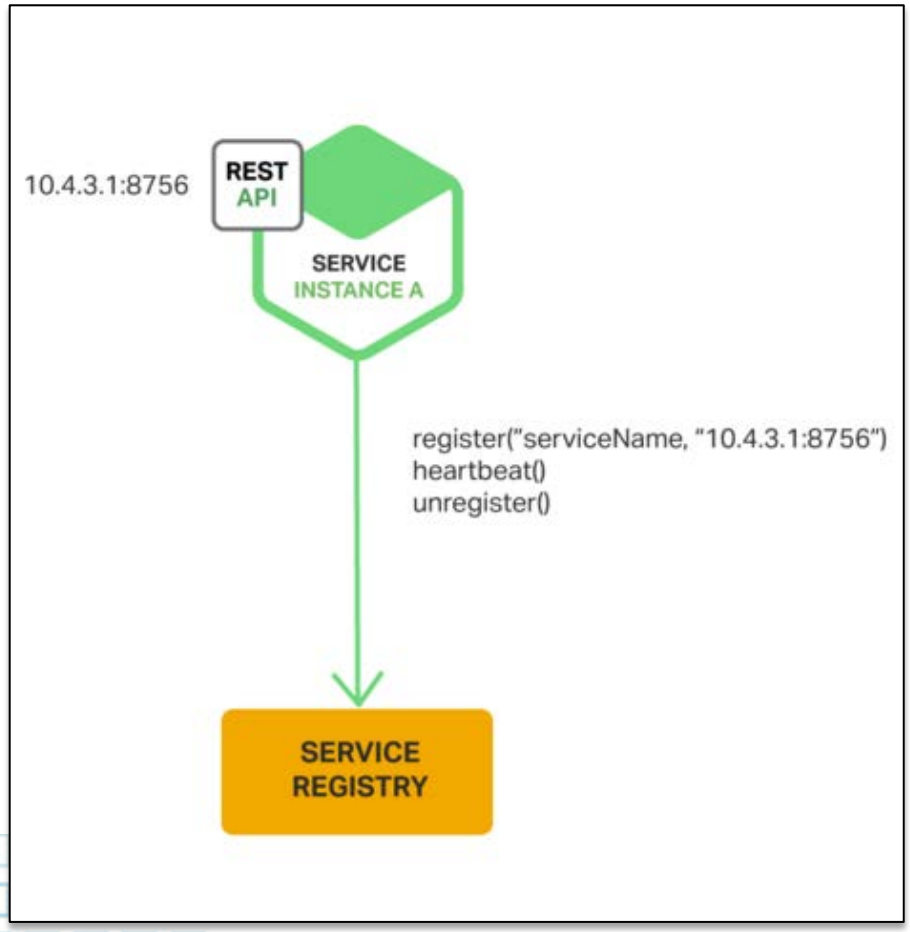
4.1 服务注册

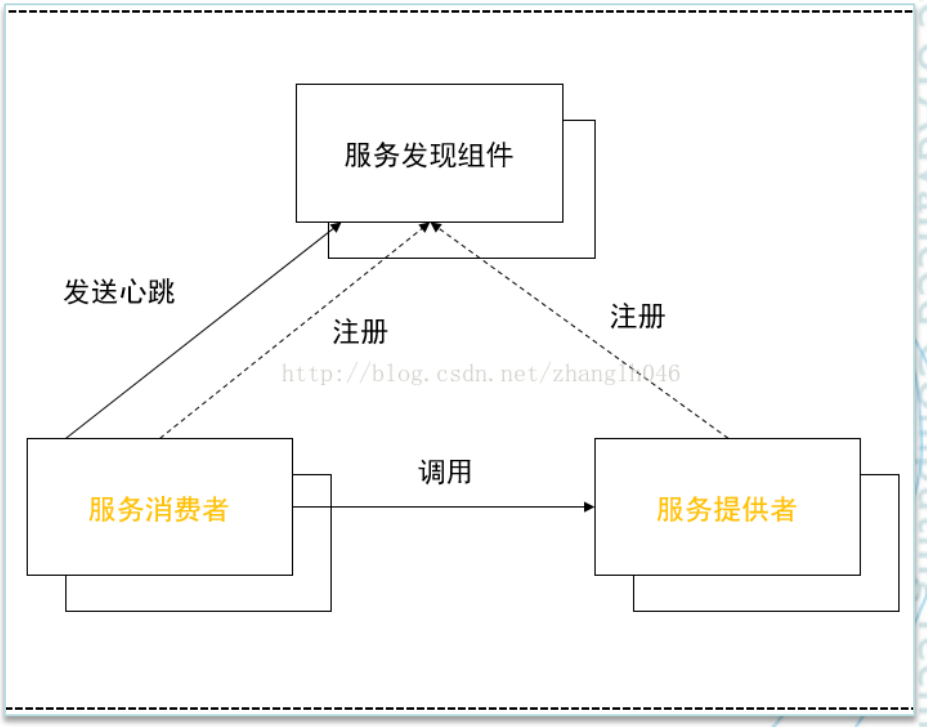
服务注册

- 将开发部署的服务信息提交到集中式的注册中心
  - 访问地址
  - 操作方法
  - 参数类型和含义
  - 服务和操作的功能描述
- 注册中心:本质上是一个逻辑集中的数据库
  - 全球集中式：UDDI
  - 企业集中式：企业UDDI和ESB
  - 平台集中式，轻量化：Zookeeper等
- 目的
  - 共享、状态监控、适配

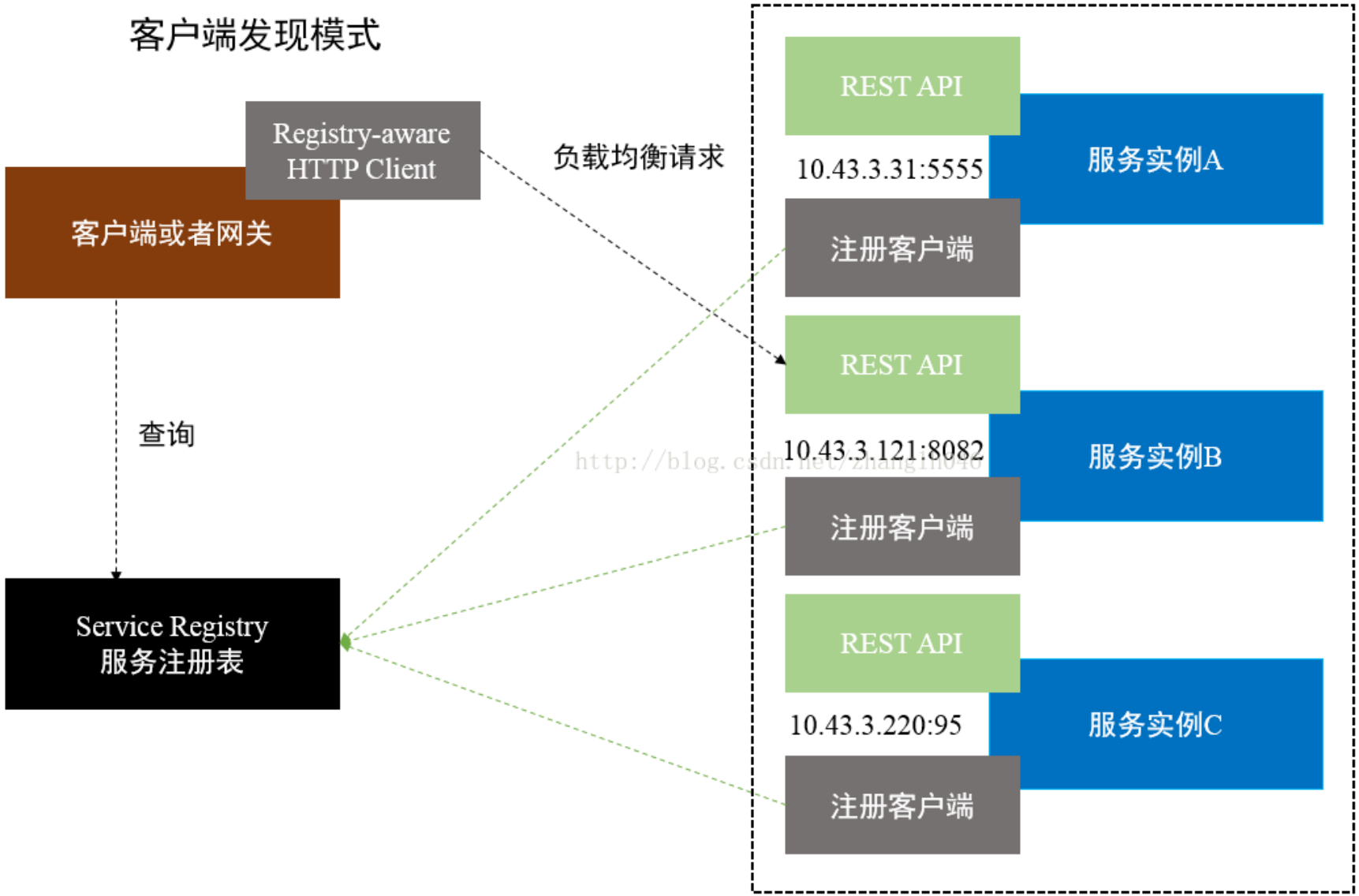
两种注册类型

- 服务自注册、服务检测组件注册





监控与适配



服务注册管理工具

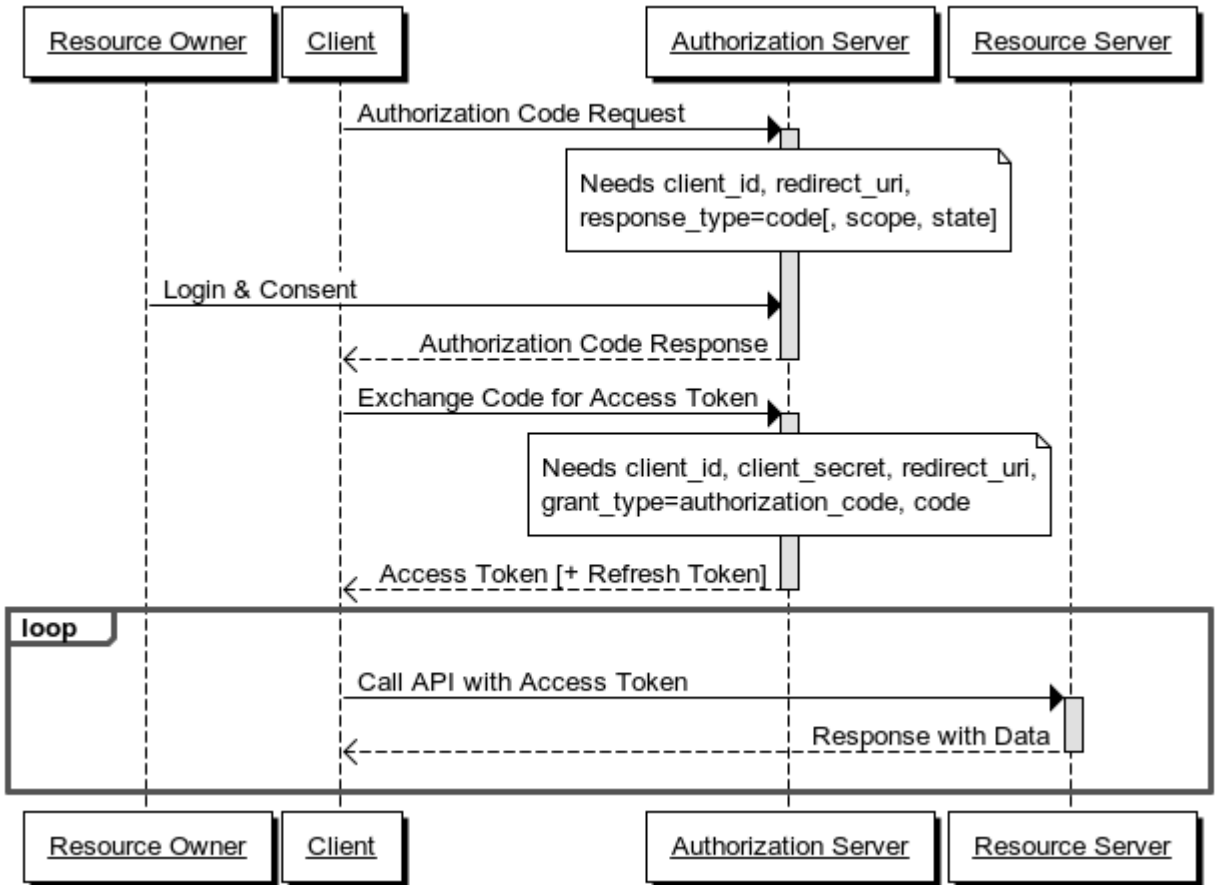
服务注册	服务部署	状态监控
Zookeeper Doozer Etcd SmartStack Eureka NSQ Serf Spotify DNS SkyDNS Consul	Cloud Foundry Gradle Docker Docker Hub Docker Machine Kitematic Docker Compose Docker Swarm AWS Jenkins Continuum Hudson Artifactory Terraform Grunt OpenShift	SonarCube Logstash New Relic Graphite Mesosphere / DCOS Winston Hystrix

服务认证

- **服务对外公开，需要身份认证与授权调用**
  - 提升服务平台的安全性
  - 服务执行需要消耗资源
  - 某些服务关联用户个人信息
- **常见认证方式：用户名/密码方式**
  - 局限性：第三方软件调用平台服务时，将获取用户在该平台上的用户名密码，比如消息分享到微博或微信
- **用户是否可以不告诉第三方软件个人用户名密码信息，但仍能授权其使用某些服务?**

OAuth2

- **Resource owner (the user),Resource server (the API), Authorization server (can be the same server as the API),Client (the third-party app)**

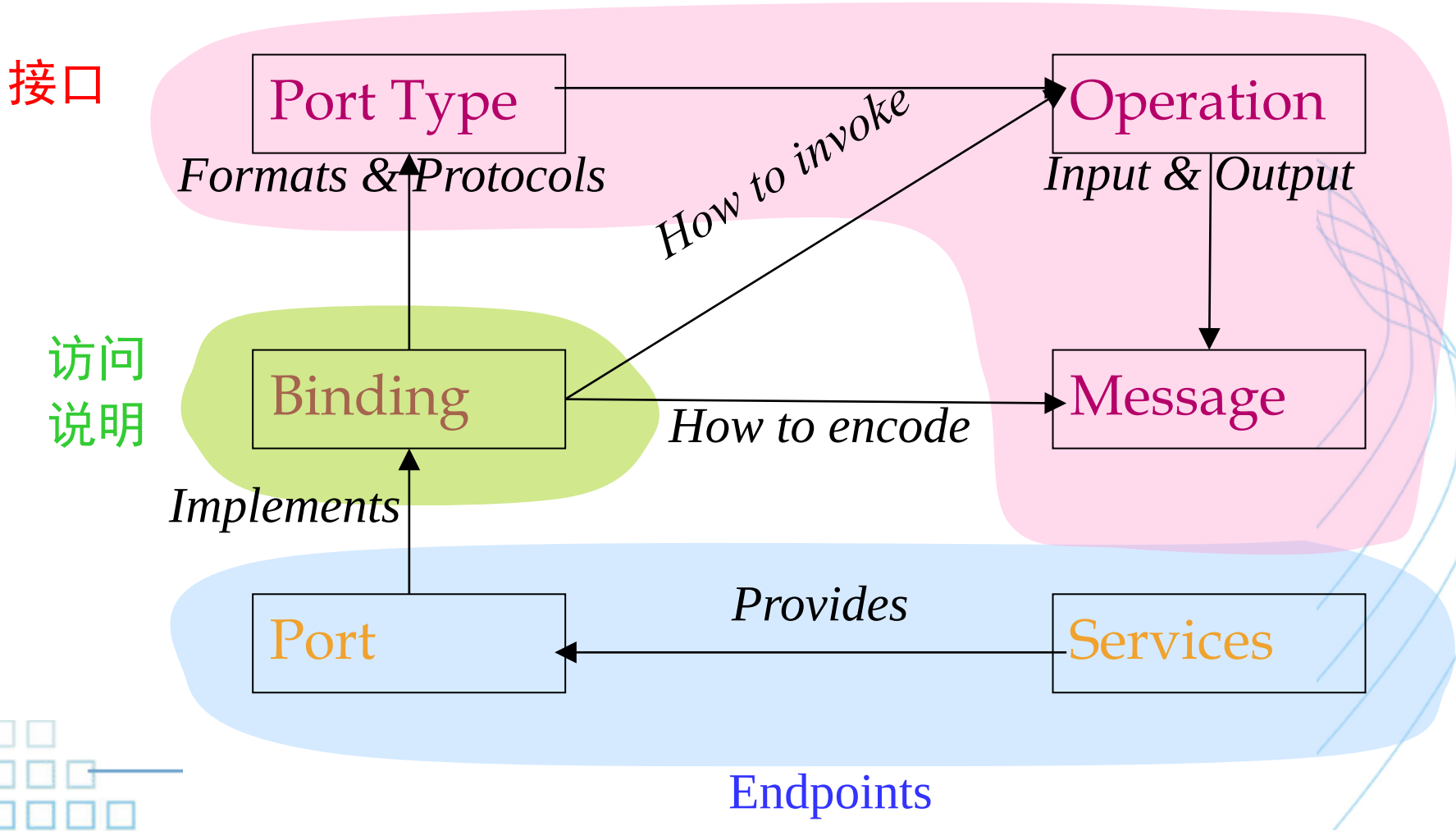


4.2 服务描述

服务描述方法

- 基本描述: WSDL
- 语义服务描述: OWL-S
- 自然语言文本描述

WSDL描述：以语法描述为主



```
<portType name="StockQuotePortType">
    <operation name="GetLastTradePrice">
      <input message="tns:GetLastTradePriceInput"/>
      <output message="tns:GetLastTradePriceOutput"/>
    </operation>
  </portType>

  <binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="GetLastTradePrice">
      <soap:operation soapAction="http://example.com/GetLastTradePrice"/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </operation>
  </binding>

  <service name="StockQuoteService">
    <documentation>My first service</documentation>
    <port name="StockQuotePort" binding="tns:StockQuoteBinding">
      <soap:address location="http://example.com/stockquote"/>
    </port>
  </service>
</definitions>
```

存在什么问题？？

Web服务技术的不足

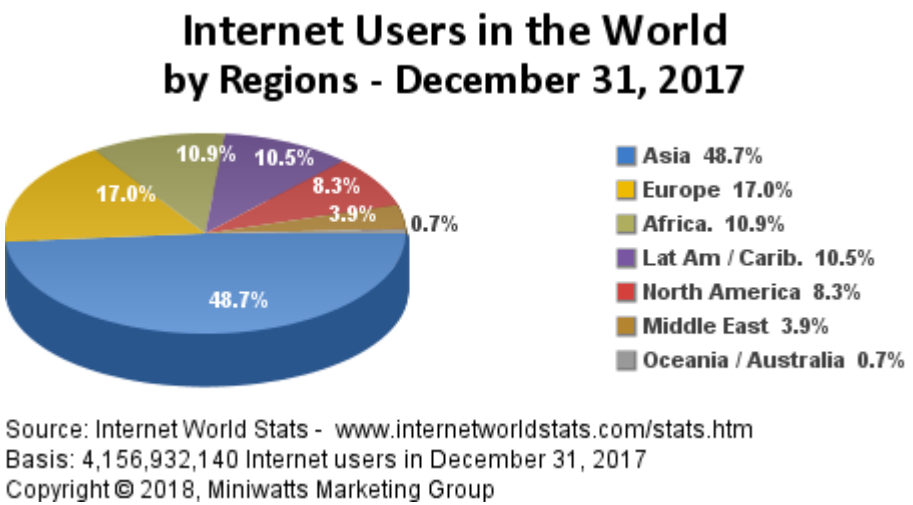
- Web服务只有语法信息描述
- 服务发现、组合和执行都基于语法信息
  - Web服务的可用性、使用、集成均需要人工检查
- 人工介入降低了可扩展性

⇒Web服务技术协议栈不能很好实现SOA/Web服务的远景目标

⇒Automate Web Service technologies

Web语义

- 借鉴Web的语义



Static

WWW

URI, HTML, HTTP

Total number of Websites

1,880,858,040

Websites online right now

The Vision

在自动信息处理中的不足：

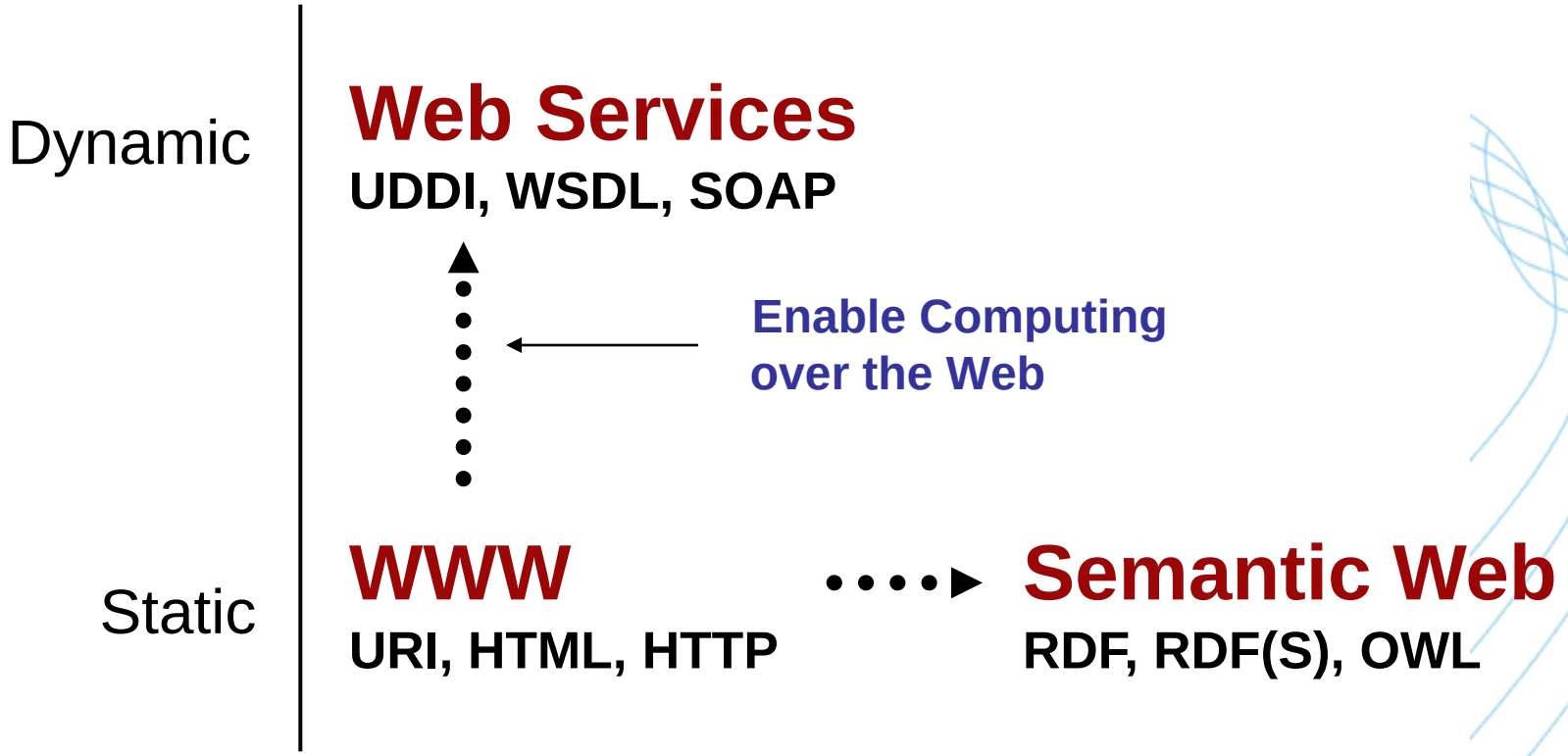
- finding
- extraction
- representation
- interpretation
- maintenance

Static

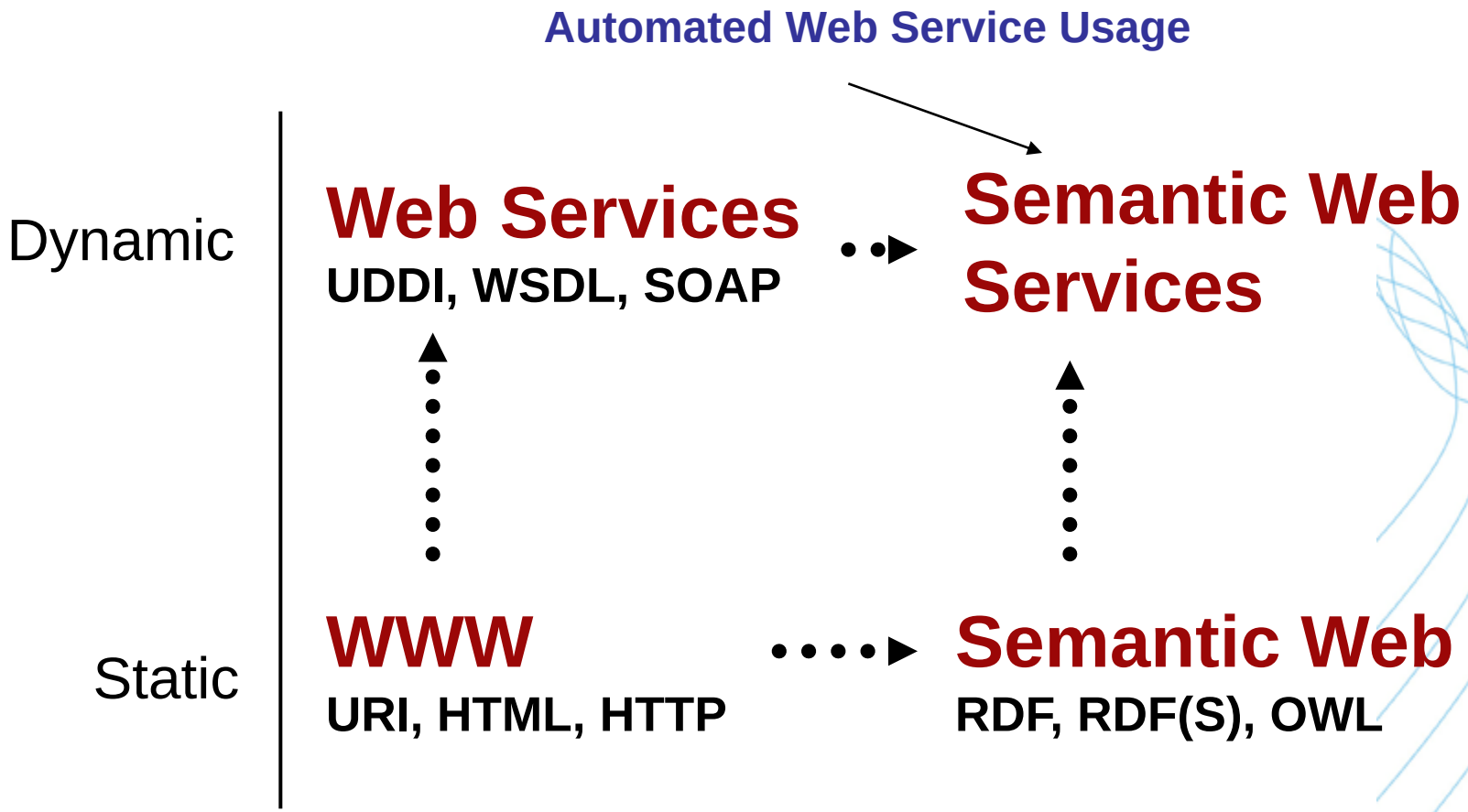
WWW ••••► **Semantic Web**

URI, HTML, HTTP **RDF, RDF(S), OWL**

The Vision



The Vision



Semantic Web

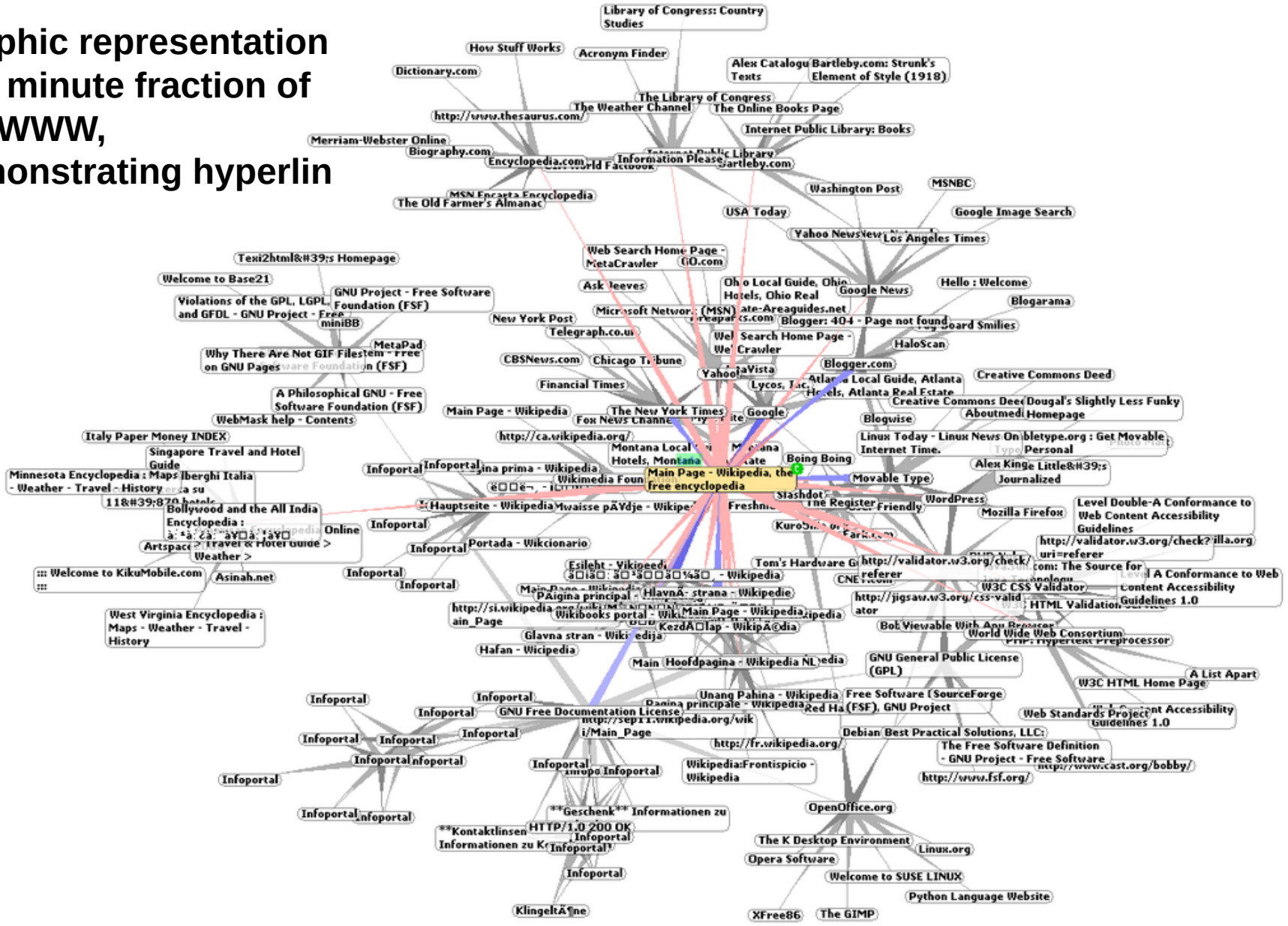
- 什么是语法Syntax、语义 Semantic
- 关于语言的意义
  - 语法、语义
  - 指示客观事物及关系
  - “今天我们在上服务计算课程”
- “神马都是浮云”、“我说的是假话”、北京和首都
- 关于语义的表达
  - 逻辑：命题逻辑、谓词逻辑

当前Web存在的问题

- WWW是最大的信息资源仓库， 包含几乎任何领域内的文档和媒体资源，并且这些数据可以在瞬间被个人和组织访问
- 其成功很大程度来自于分布式设计，即Web页可以存放于任何一台主机，通过超链可以访问本机或远程的页面
- Web存在的问题
  - 信息难以抽取：
    - 具有无限的潜力，然而尚未发挥出来，困难在于Web页上的信息内容很难抽取
  - 资源难以定位：
    - Web的大小使得很难定位相关的信息资源
  - 推理难以实施：
    - 目录服务（Yahoo）和搜索引擎（Google）提供了一些帮助，但远不能满足用户的需求
    - 用户难以让Web作更多的、功能远远超过目录和搜索的事情，比如让Web为用户安排一个完美的度假
- 上述问题的根源在于：Web不是设计给机器处理的

Web的展现

Graphic representation of a minute fraction of the WWW, demonstrating hyperlinks

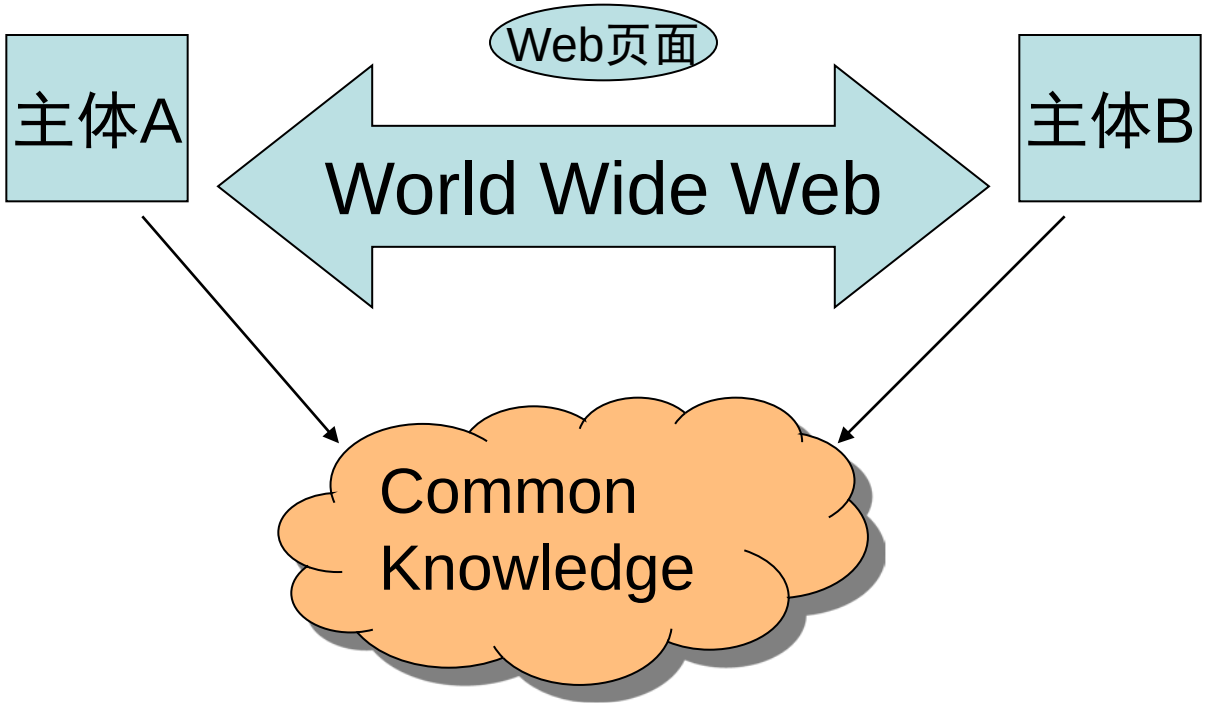


Web Page

```
1481        <!-- 1 -->
1482  <div class="b_cont" id="blk_jdzt_01">
1483
1484  <div class="ct_pt_01 clearfix">
1485      <div class="ct_pic"><a href="http://news.sina.com.cn/z/rbgd2012/" target="_blank"><img
      src="http://i0.sinaimg.cn/dy/2012/0925/U6606P1DT20120925114305.jpg" width="90" height="60" alt="" /></a></div>
1486      <div class="ct_txt">
1487        <h4><a href="http://news.sina.com.cn/z/rbgd2012/" target="_blank">中日钓鱼岛争端升级</a></h4>
1488        <p>中国宣布钓鱼岛领海基线并派公务船巡航。<a href="http://news.sina.com.cn/z/rbgd2012/"
      target="_blank">[详细]</a></p>
1489      </div>
1490    </div>
1491
1492  <div class="ct_pt_01 clearfix">
1493      <div class="ct_pic"><a href="http://mil.news.sina.com.cn/nz/zgdyshm/" target="_blank"><img
      src="http://i3.sinaimg.cn/dy/2012/0925/U6606P1DT20120925114551.jpg" width="90" height="60" alt="" /></a></div>
1494      <div class="ct_txt">
1495        <h4><a href="http://mil.news.sina.com.cn/nz/zgdyshm/" target="_blank">我国首艘航母辽宁号服役</a></h4>
1496        <p>9月25日，我国首艘航母辽宁号入列服役。<a href="http://mil.news.sina.com.cn/nz/zgdyshm/"
      target="_blank">[详细]</a></p>
1497      </div>
1498    </div>
1499
1500    <ul class="list_12 link_c666">
1501
1502      <li><a href="http://news.sina.com.cn/z/2012mgdx/" target="_blank">2012年美国大选</a></li>
1503
1504      <li><a href="http://news.sina.com.cn/pc/2012-09-04/27/7328.html" target="_blank">2012边疆之星年度人物评
      选</a></li>
1505
1506    </ul>
1507  </div>
```

当前Web上的通讯

- 主体A把信息放到网页中，Web在主体B的浏览器端显示出来，实际上仍然是人之间的通讯，Web并不理解网页中的内容

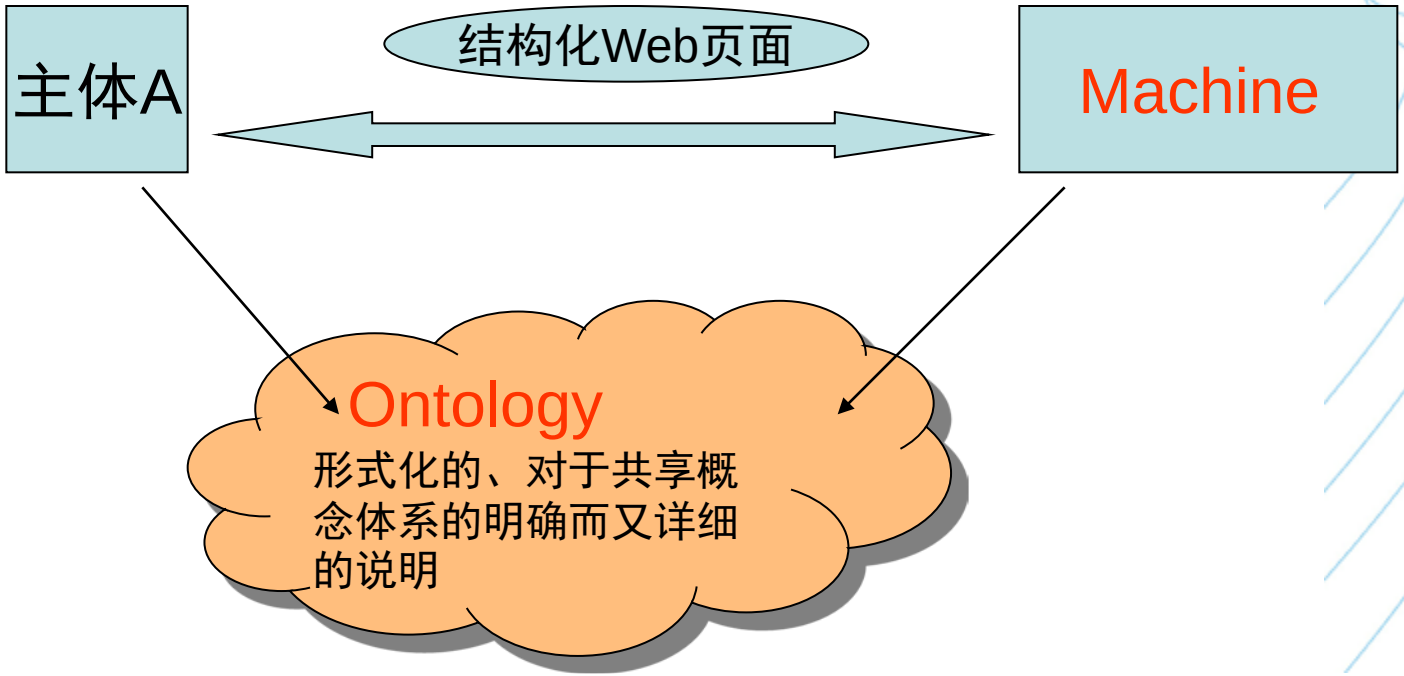


如何让机器理解Web

- 两种途径：
- 1. 自然语言理解技术，然而仍然有很多关键问题没有解决
- 2. 用知识表示语言来描述Web页，即构造一种新的Web——语义Web

语义网上的通讯

- Web携带语义信息，使机器能够理解Web页面，从而实现强大的功能。
- 需要一个人和机器都能理解的Ontology



为什么需要 Ontology

- Ontology为人类和应用程序系统提供了一个对于主题的共同理解
- Ontology为了不同来源的信息的合成，提供了一个共同的相关领域的理解
- Ontology为了在不同的应用程序之间共享信息和知识（用于互操作），描述应用程序的领域，定义术语及其关系
  - 提供机器能够理解的信息结构
  - 使领域知识能够被再利用

Ontology的概念

- 最早是一个哲学概念，从哲学的范畴来说，Ontology 是客观存在的一个系统的解释或说明，关心的是客观现实的抽象本质
- 在人工智能界，Ontology:“给出构成相关领域词汇的基本术语和关系，以及利用这些术语和关系构成的规定这些词汇外延的规则的定义”。
- 通常认为,Ontology是共享概念模型的明确的形式化规范说明

Ontology-本体

- 本体提供的是一种共享词表，也就是特定领域之中那些存在着的对象类型或概念及其属性和相互关系
- 本体就是一种特殊类型的术语集，具有结构化的特点，且更加适合于在计算机系统之中使用
- 本体实际上就是对特定领域之中某套概念及其相互之间关系的形式化表达
- 分为上层本体和领域本体
  - Web Ontology – OWL
  - COSMO：属于一部基础本体（当前版本为OWL），其设计旨在收录所有从逻辑上说明任何领域实体的含义时所需的原初型概念
  - 有关内科学与外科学医学术语的GALEN本体（OWL-DL格式）
  - 基因组学领域的基因本体（Gene Ontology,GO）
  - 疾病本体（Disease Ontology）：在设计上旨在促进各种疾病及相关健康状况向特定医学代码的映射。

参阅：
http://zh.wikipedia.org/wiki/%E6%9C%AC%E4%BD%93_(%E4%BF%A1%E6%81%AF%E7%A7%91%E5%AD%A6)

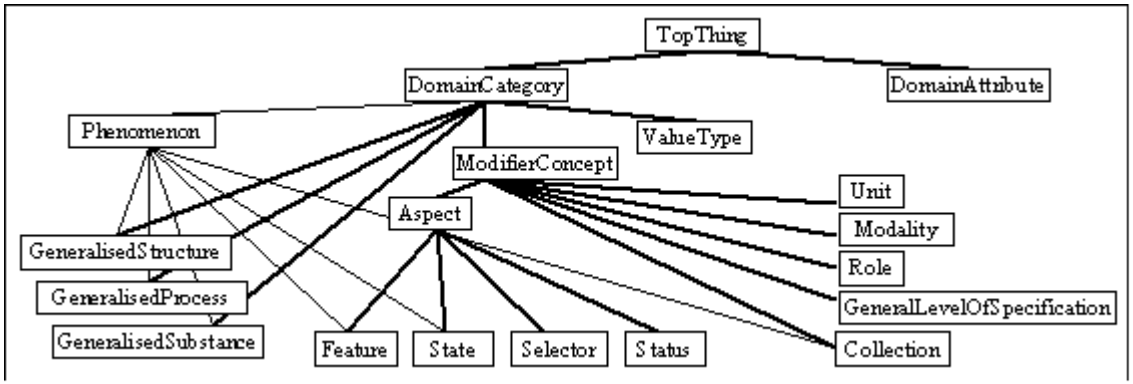
Programming Languages

- 法国人（不懂中文）
- 中国人（不懂法文）
- 计算机（既不懂中文，也不懂法文）

- 为什么都能看懂Programming Language
  - Java、C/C++、Python……
  - PL的语法、语义，定义了一套关键字和句子
  - 是中国人、法国人、计算机都认可、都理解的

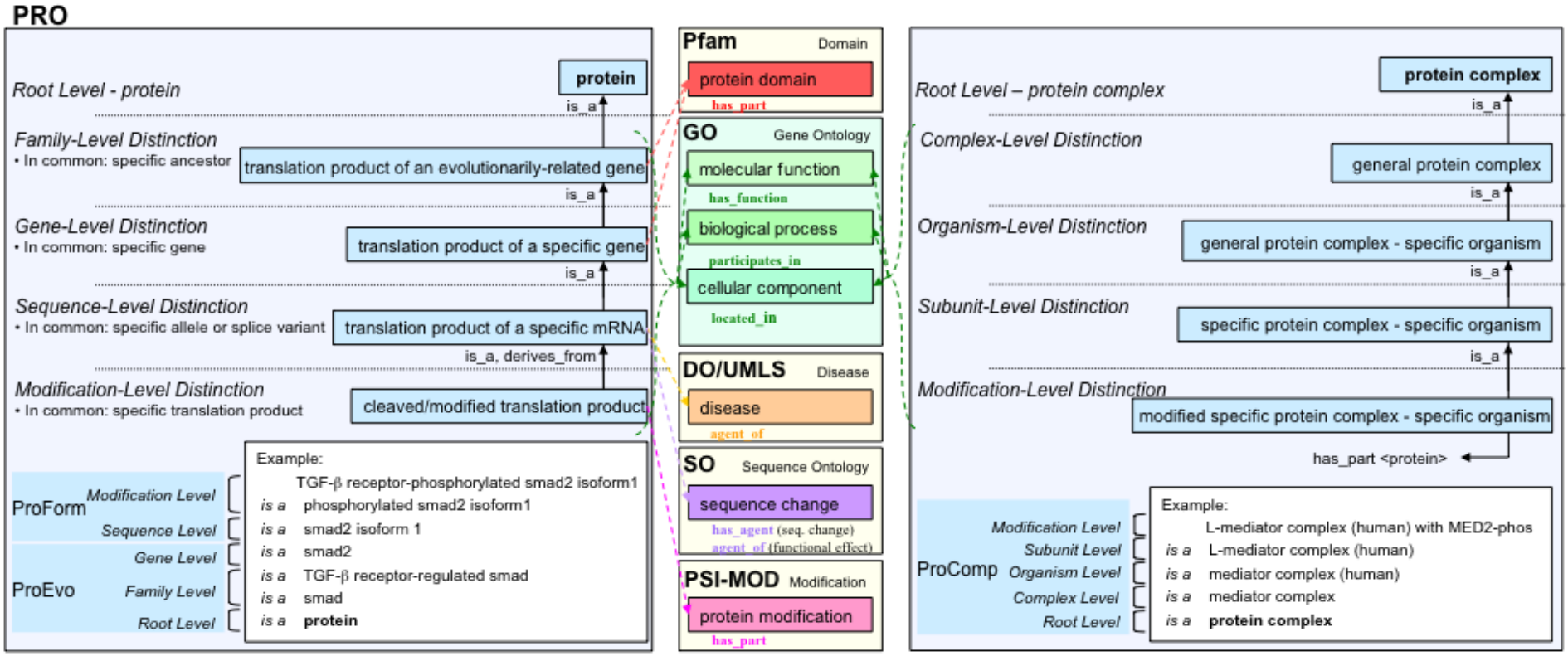
GALEN 医学本体

- GALEN是医学中的语言、百科全书和命名法的普遍体系结构，它使用称为 GRAIL 的一个描述逻辑提供临床应用的语言、术语、和编码服务。



http://202.38.126.151/index.php/GALEN_%E5%8C%BB%E5%AD%A6%E6%9C%AF%E8%AF%AD%E4%B8%8E%E5%8C%BB%E5%AD%A6%E6%9C%AC%E4%BD%93

乔治敦大学 蛋白质本体PRO



Ontology实例

Concept

conceptual entity of the domain

Property

attribute describing a concept

Relation

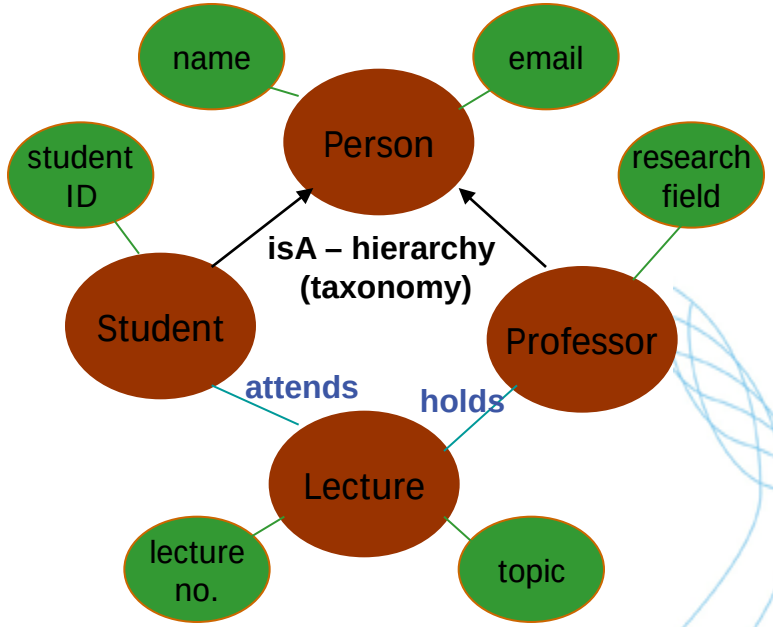
relationship between concepts or properties

Axiom

coherency description between Concepts / Properties / Relations via logical expressions

Instance

individual in the domain

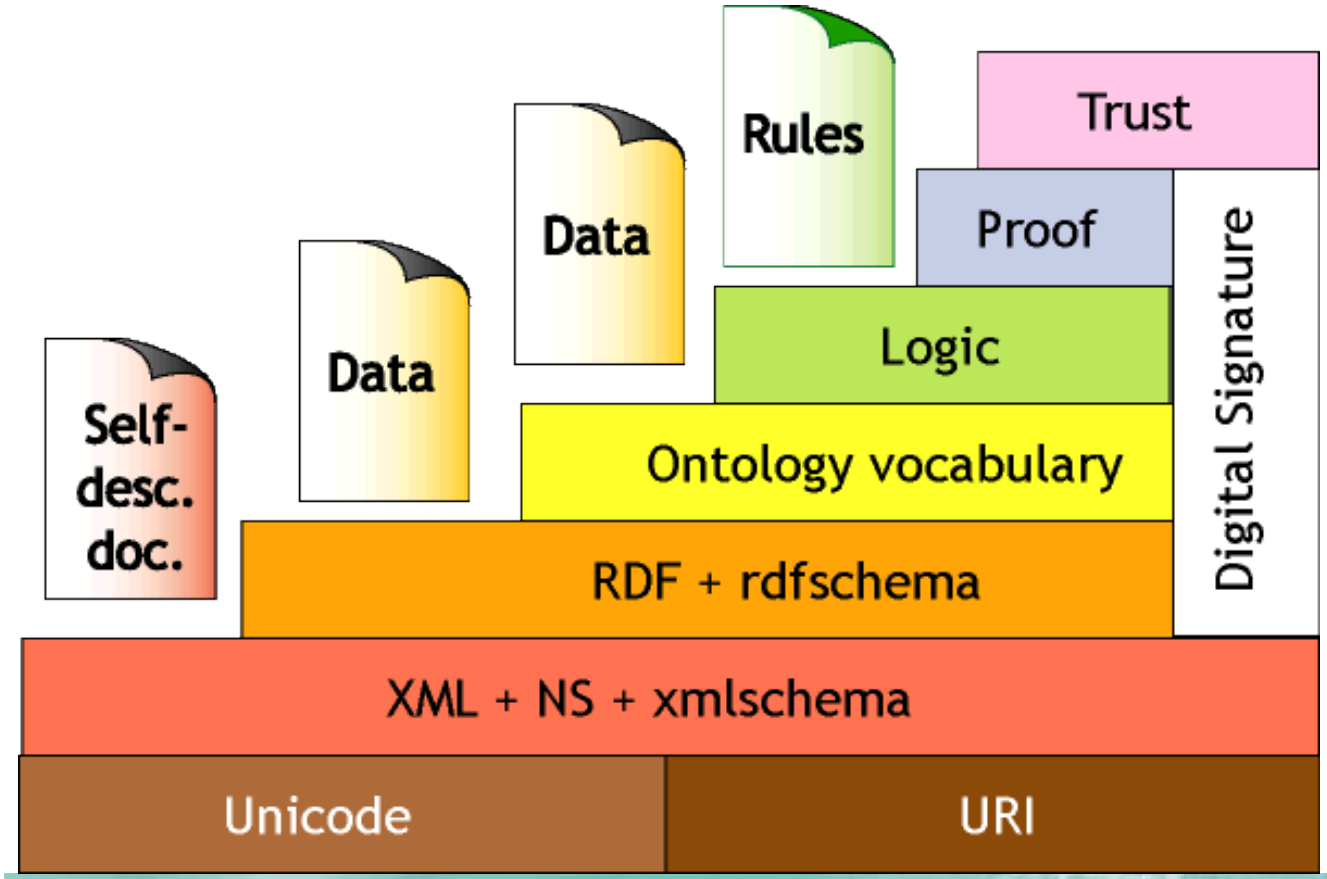


holds(Professor, Lecture) =>
Lecture.topic = Professor.researchField

Ann memberOf student
name = Ann Lee
studentID = 12345

语义Web(Semantic Web)

- 通过给万维网上的文档(如:HTML)添加能够被计算机所理解的语义(Meta data)，从而使整个互联网成为一个通用的信息交换媒介。



Web Ontology Language OWL

- Semantic Web led to requirement for a "web ontology language"
- W3C set up Web-Ontology (WebOnt) Working Group
  - **WebOnt developed OWL language**
  - **OWL based on earlier languages OIL and DAML+OIL**
  - **OWL now a W3C recommendation (i.e., a standard)**
- OIL, DAML+OIL and OWL based on Description Logics
  - **OWL effectively a "Web-friendly" syntax for SHOIN**

OWL RDF/XML Exchange Syntax

$$Person \sqcap \forall hasChild.(Doctor \sqcup \exists hasChild.Doctor)$$

```
<owl:Class>
  <owl:intersectionOf rdf:parseType=" collection">
    <owl:Class rdf:about="#Person"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasChild"/>
      <owl:allValuesFrom>
        <owl:unionOf rdf:parseType=" collection">
          <owl:Class rdf:about="#Doctor"/>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hasChild"/>
            <owl:someValuesFrom rdf:resource="#Doctor"/>
          </owl:Restriction>
        </owl:unionOf>
      </owl:allValuesFrom>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

医生世家：所有孩子中，要么自己是医生，要么自己的孩子中也有当医生的

Class/Concept Constructors

Constructor	DL Syntax	Example	FOL Syntax
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Human ⊓ Male	$C_1(x) \wedge \ldots \wedge C_n(x)$
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor ⊔ Lawyer	$C_1(x) \vee \ldots \vee C_n(x)$
complementOf	$\neg C$	¬Male	$\neg C(x)$
oneOf	$\{x_1\} \sqcup \ldots \sqcup \{x_n\}$	{john} ⊔ {mary}	$x = x_1 \vee \ldots \vee x = x_n$
allValuesFrom	$\forall P.C$	∀hasChild.Doctor	$\forall y.P(x,y) \rightarrow C(y)$
someValuesFrom	$\exists P.C$	∃hasChild.Lawyer	$\exists y.P(x,y) \wedge C(y)$
maxCardinality	$\leqslant nP$	⩽1hasChild	$\exists^{\leqslant n} y.P(x,y)$
minCardinality	$\geqslant nP$	⩾2hasChild	$\exists^{\geqslant n} y.P(x,y)$

- **C is a concept (class); P is a role (property); x_i is an individual/nominal**
- **XMLS datatypes as well as classes in ∀P.C and ∃P.C**
  - **Restricted form of DL concrete domains**

Ontology Axioms

OWL Syntax	DL Syntax	Example
subClassOf	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
equivalentClass	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
subPropertyOf	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
equivalentProperty	$P_1 \equiv P_2$	cost $\equiv$ price
transitiveProperty	$P^+ \sqsubseteq P$	ancestor$^+$ $\sqsubseteq$ ancestor

OWL Syntax	DL Syntax	Example
type	$a : C$	John : Happy-Father
property	$\langle a, b \rangle : R$	$\langle$John, Mary$\rangle$: has-child

- **OWL ontology equivalent to DL KB (Tbox + Abox)**

语义Web服务

- 自动化的Web服务技术，基于以下技术
  - Web服务的形式标注
  - 基于推理的(半)自动化的服务发现、组合、调解、执行
- 集成语义Web技术
  - 将Ontology作为数据模型

Semantic Web

- 数据的机器解释
- 本体作为数据模型

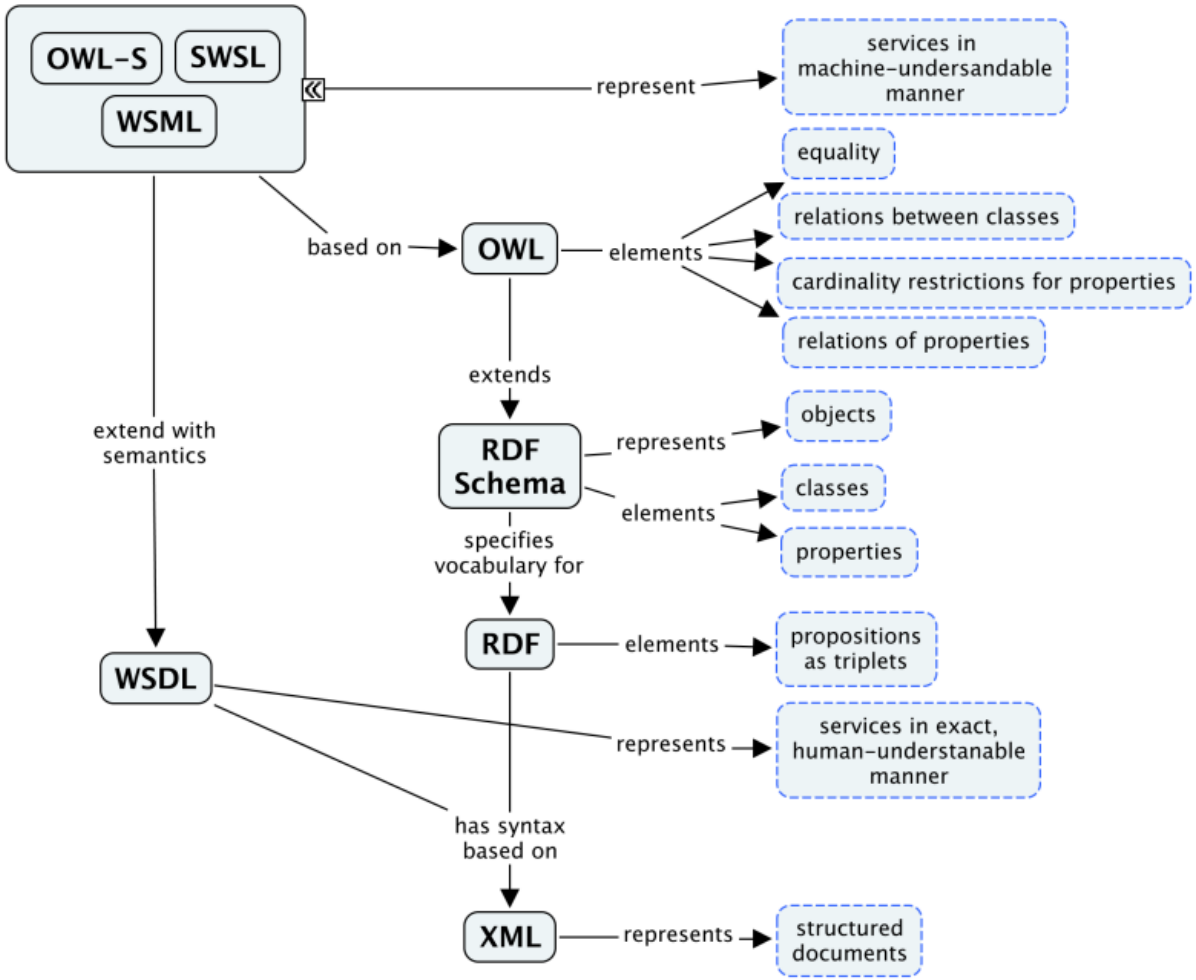
+

服务的自动发现、选择、组合和执行

Web Service

语义Web服务是实现下一代Web远景目标的集成方案

语义Web相关语言的关系



Semantic Web Services Language (SWSL) http://www.w3.org/Submission/SWSF-SWSL/
Web Service Modeling Language (WSML) http://www.w3.org/Submission/WSML/
OWL-S: Semantic Markup for Web Services http://www.w3.org/Submission/OWL-S/
Resource Description Framework (RDF) http://www.w3.org/RDF/

Web服务使用过程

1. Deployment 创建、发布Web服务描述
2. Discovery 决定对一个请求的可使用服务
3. Composition 组装服务，达到目标
4. Selection 在可用服务集中选择一个最合适的服务
5. Mediation 解决阻碍互操作的失配（数据、协议、过程）
6. Execution 调用服务

语义Web服务可有效支持对上述过程

Web服务的语义类型

- **信息模型**的语义：服务使用、暴露的数据的语义
- **功能**语义：服务做什么
- **非功能**语义：关于服务非功能性属性的语义，如QoS、安全等
- **行为**语义：关于服务行为的语义，比如choreography, faults等

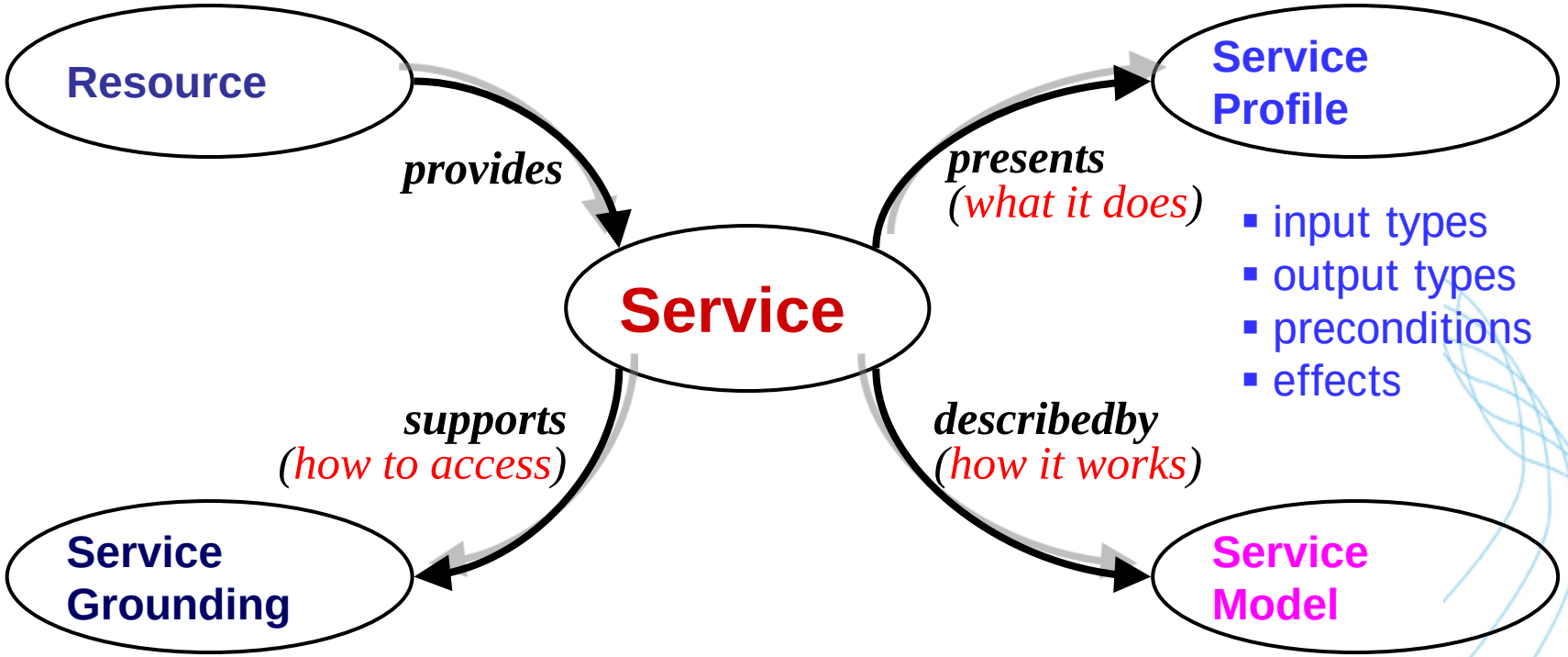
语义Web服务的标准规范

- Web Services
  - Top-down: OWL-S, WSMO
  - Bottom-up: WSDL-S/SAWSDL, WSMO-Lite
- OWL-S: Semantic Markup for Web Services
  - Ontology Web Language for Services

OWL-S

- 一个上层ontology，用于描述：
  - Web服务的properties & capabilities
- Automation Enabled by OWL-S
  - Web Service Discovery & Selection
    - 发现飞往纽约的航班
  - Web Service Invocation
    - 预订18日抵达的美联航机票
  - Web Service Composition & Interoperation
    - 安排从圣巴巴拉、经纽约、到波士顿的航班、出租车、酒店
  - Web Service Execution Monitoring
    - 去机场的出租车是否已经预订成功？

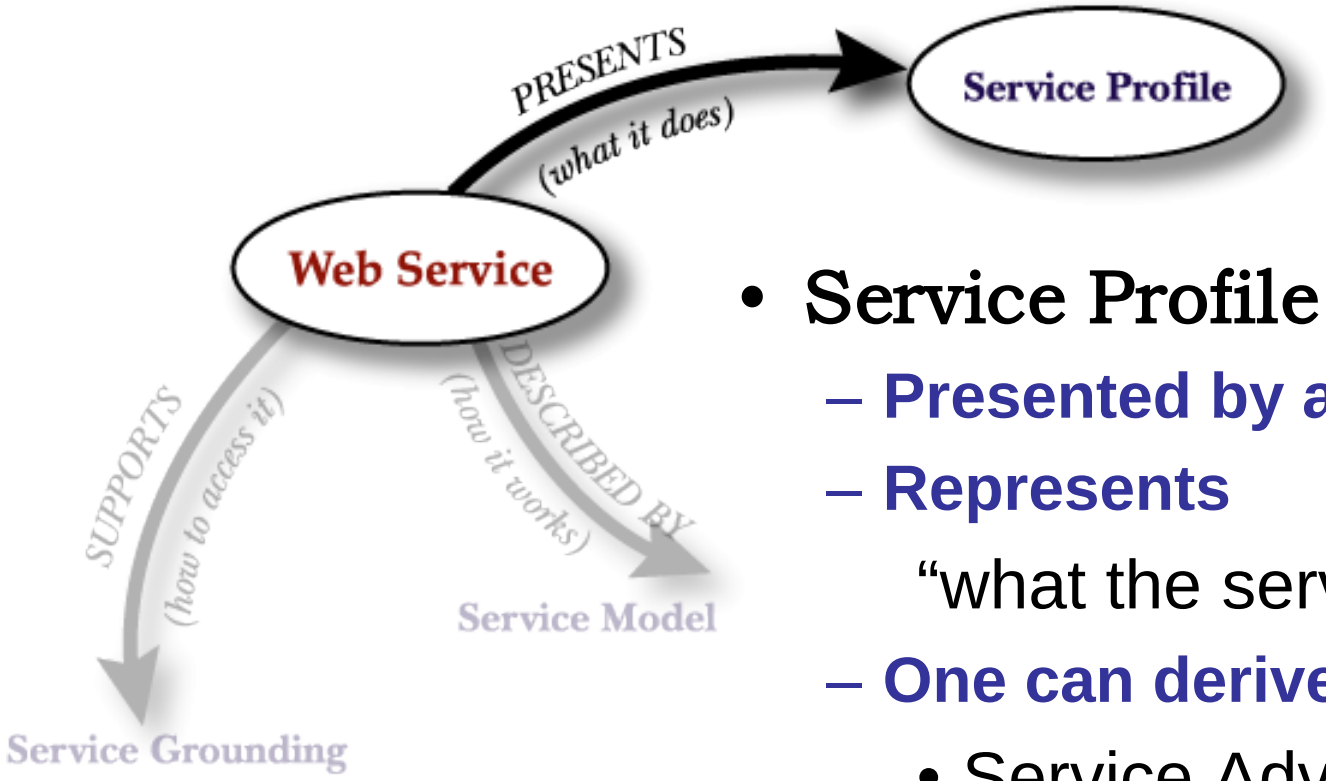
OWL-S 顶层Ontology



- communication protocol (RPC, HTTP, ...)
- port number
- marshalling/serialization

- process flow
- composition hierarchy
- process definitions

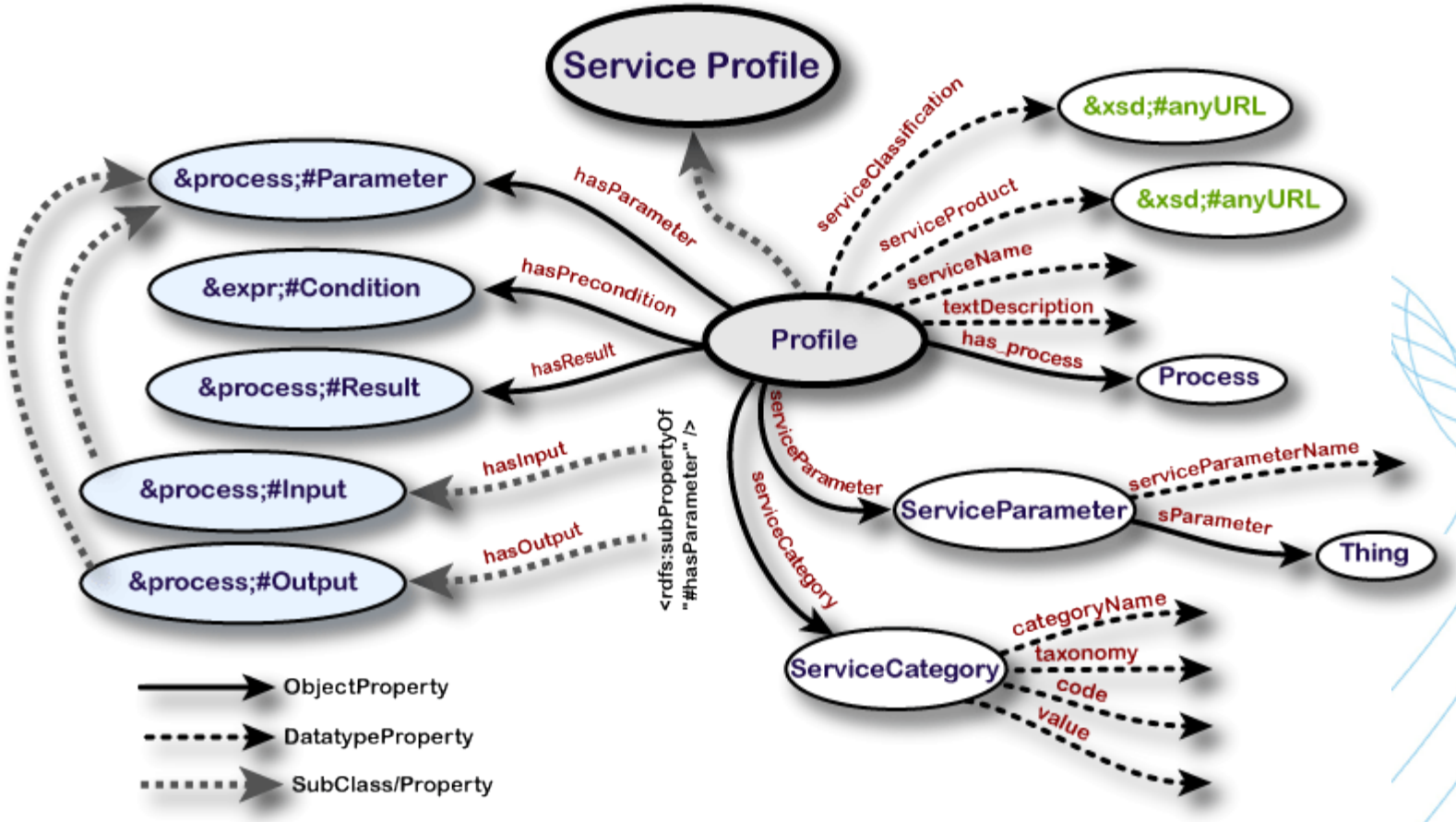
Service Profiles



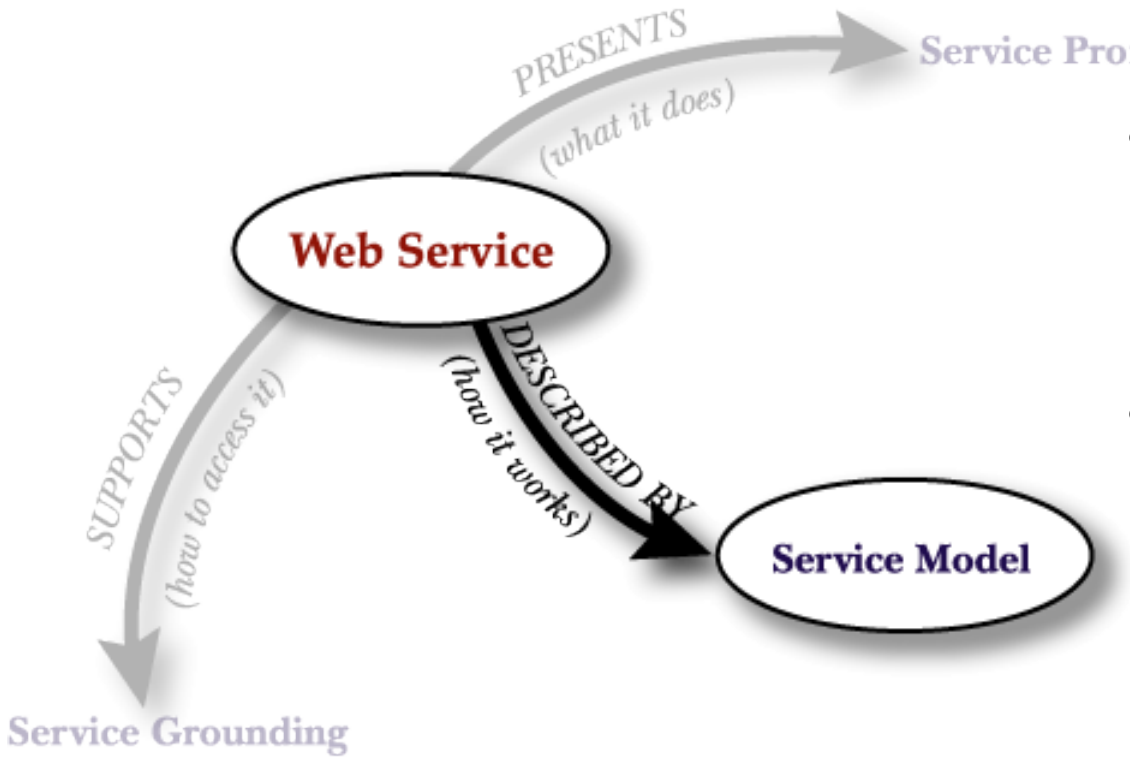
- Service Profile
  - **Presented by a service**
  - **Represents**

    "what the service provides"
  - **One can derive:**
    - Service Advertisements
    - Service Requests

OWL-S Service Profile



Service Models



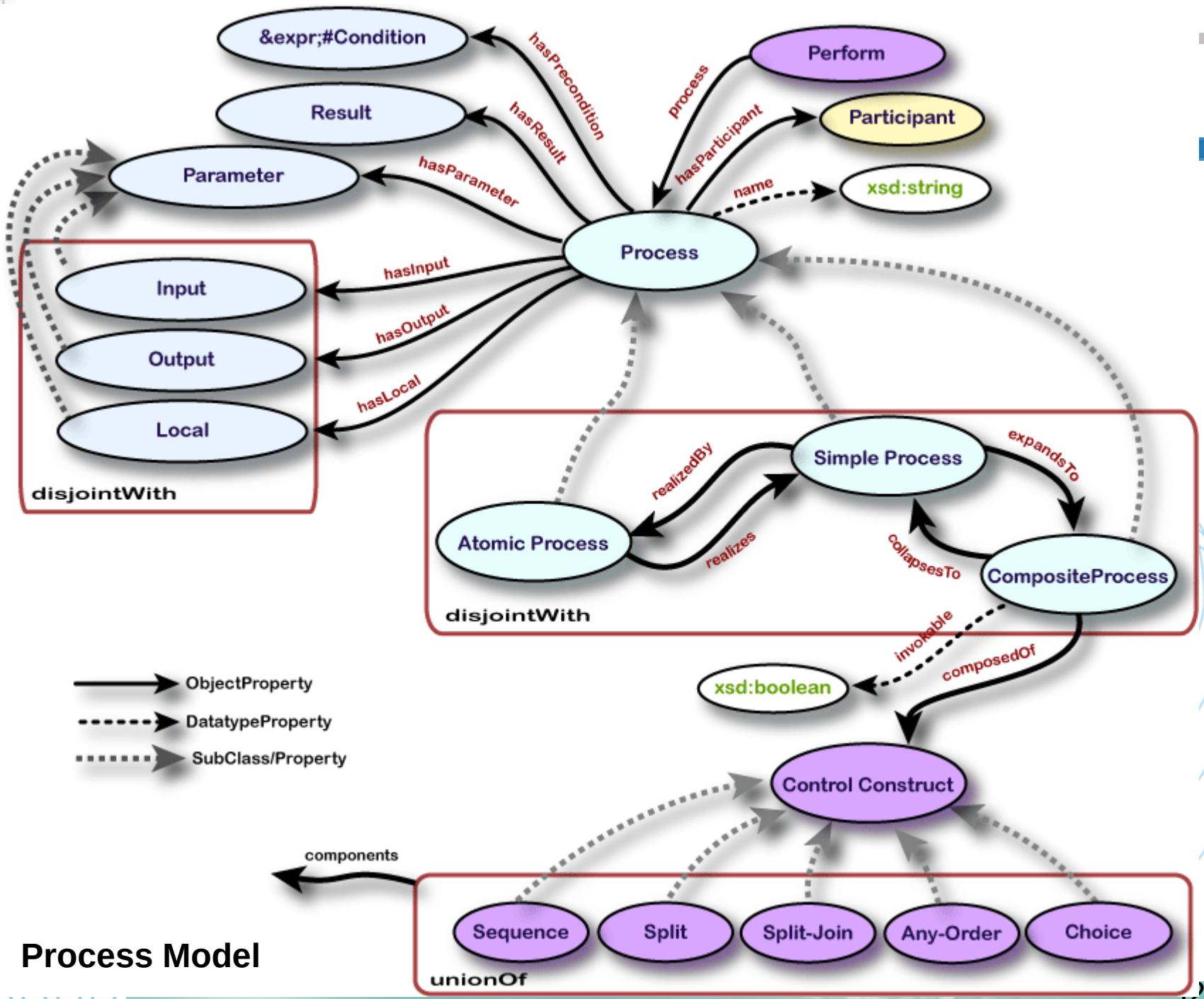
- Service Process
  - Describes how a service works
- Facilitates
  - (automated) Web service invocation
  - composition
  - interoperation
  - monitoring

The Process Ontology

- Process Ontology 的基本类是Process.
- 包括:
  - any number of (possibly, conditional) inputs;
  - any number of (possibly, conditional) outputs;
  - any number of preconditions, which must hold in order for the process to be invoked;
  - any number of (possibly, conditional) side effects;
  - any number of participants (subprocess)

Process的类型

- Atomic processes: 直接可调用、无子过程、单步执行
- Composite processes: 由一组其他组合/非组合过程构成，包含composedOf 属性，表明该组合过程的控制结构(ControlConstruct 子类)
- Simple processes: 一些抽象概念，用于提供某些原子过程的一个视图，某些组合过程的一个简化表达（黑盒）

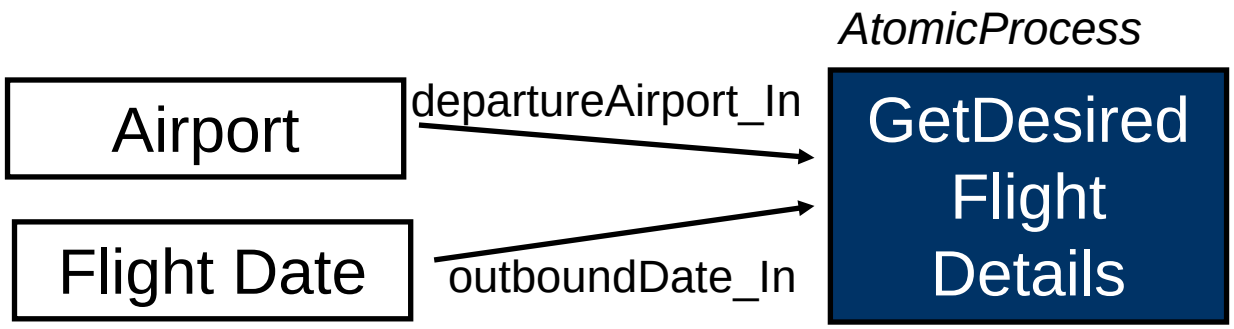


Process Model

过程的控制结构

Construct	Description
Sequence	Execute a list of processes in a sequential order
Concurrent	Execute elements of a bag of processes concurrently
Split	Invoke elements of a bag of processes
Split+Join	Invoke elements of a bag of processes and synchronize
Unordered	Execute all processes in a bag in any order
Choice	Choose between alternatives and execute one
If-then-else	If specified condition hold, execute “Then”, else execute “Else”.
Repeat-Until	Iterate execution of a bag of processes until a condition holds
Repeat-While	Iterate execution of a bag of processes while a condition holds

Atomic Process Example



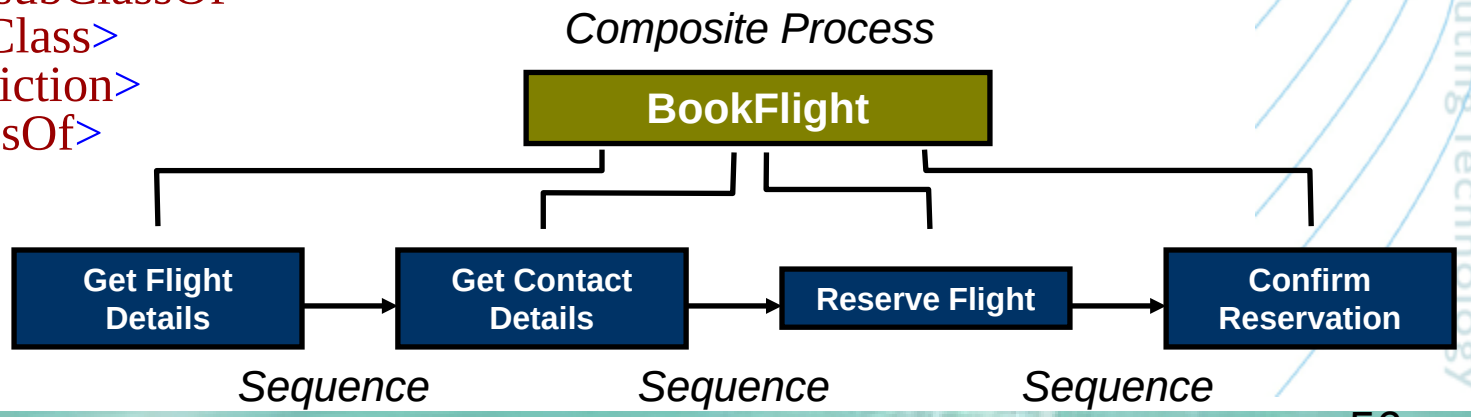

```
<!– Atomic Process Definition - GetDesiredFlightDetails  -->
<rdfs:Class rdf:ID="GetDesiredFlightDetails" >
    <rdfs:subClassOf rdf:resource=
        "http://www.daml.org/Process#AtomicProcess"
/>
</rdfs:Class>
```

Atomic Process Example

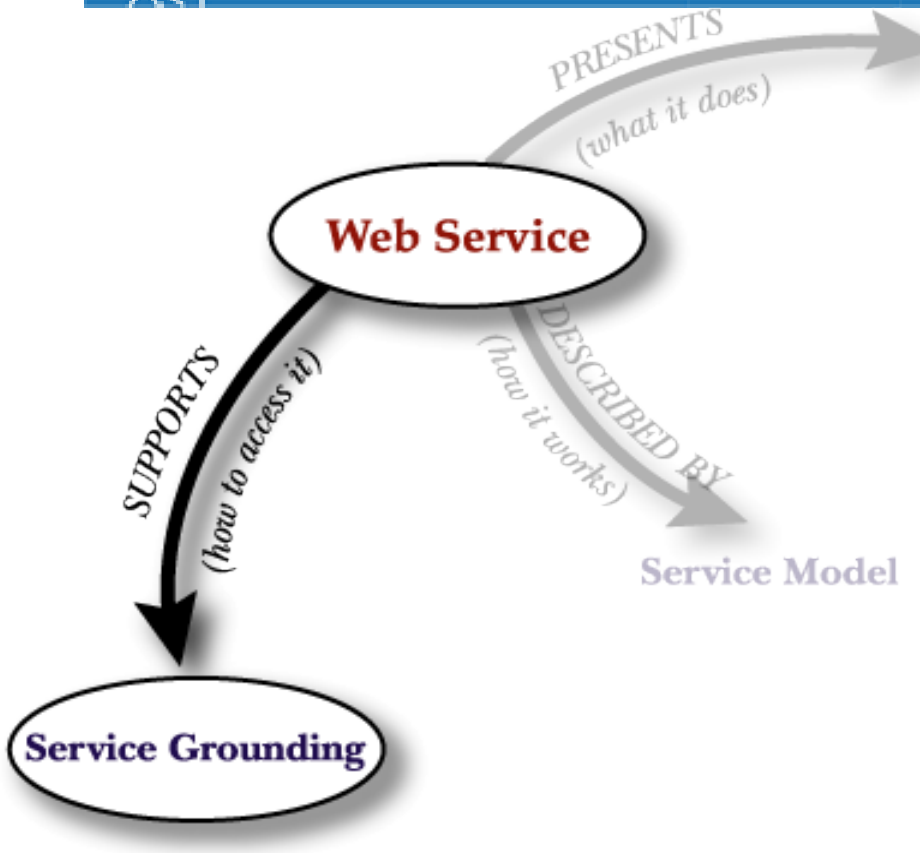
```
<!– (sample) Inputs used by atomic process GetDesiredFlightDetails -->
<rdf:Property rdf:ID="departureAirport_In">
    <rdfs:subPropertyOf rdf:resource=
            "http://www.daml.org/Process#input" />
    <rdfs:domain rdf:resource="#GetDesiredFlightDetails" />
    <rdfs:range rdf:resource="http://www.daml.ri.cmu.edu/ont/
      DAML-S/concepts.daml#Airport" />
</rdf:Property>

<rdf:Property rdf:ID="outboundDate_In">
    <rdfs:subPropertyOf rdf:resource="http://www.daml.org/Process#input" />
    <rdfs:domain rdf:resource="#GetDesiredFlightDetails" />
    <rdfs:range rdf:resource="http://www.daml.ri.cmu.edu/ont/
          DAML-S/concepts.daml#FlightDate" />
  </rdf:Property>
```

Composite Process Example

```
<rdfs:Class rdf:ID="BookFlight">
  <rdfs:subClassOf rdf:resource="#CompositeProcess" />
  <rdfs:subClassOf rdf:resource="http://www.daml.org/Process#Sequence" />
  <daml:subClassOf>
    <daml:Restriction>
      <daml:onProperty rdf:resource="http://www.daml.org/Process#components" />
      <daml:toClass>
        <daml:subClassOf>
          <daml:unionOf rdf:parseType="daml:collection">
            <rdfs:Class rdfs:about="#GetFlightDetails" />
            <rdfs:Class rdfs:about="#GetContactDetails" />
            <rdfs:Class rdfs:about="#ReserveFlight" />
            <rdfs:Class rdfs:about="#ConfirmReservation" />
          </daml:unionOf>
        </daml:subClassOf>
      </daml:toClass>
    </daml:Restriction>
  </daml:subClassOf>
</rdfs:Class>
```

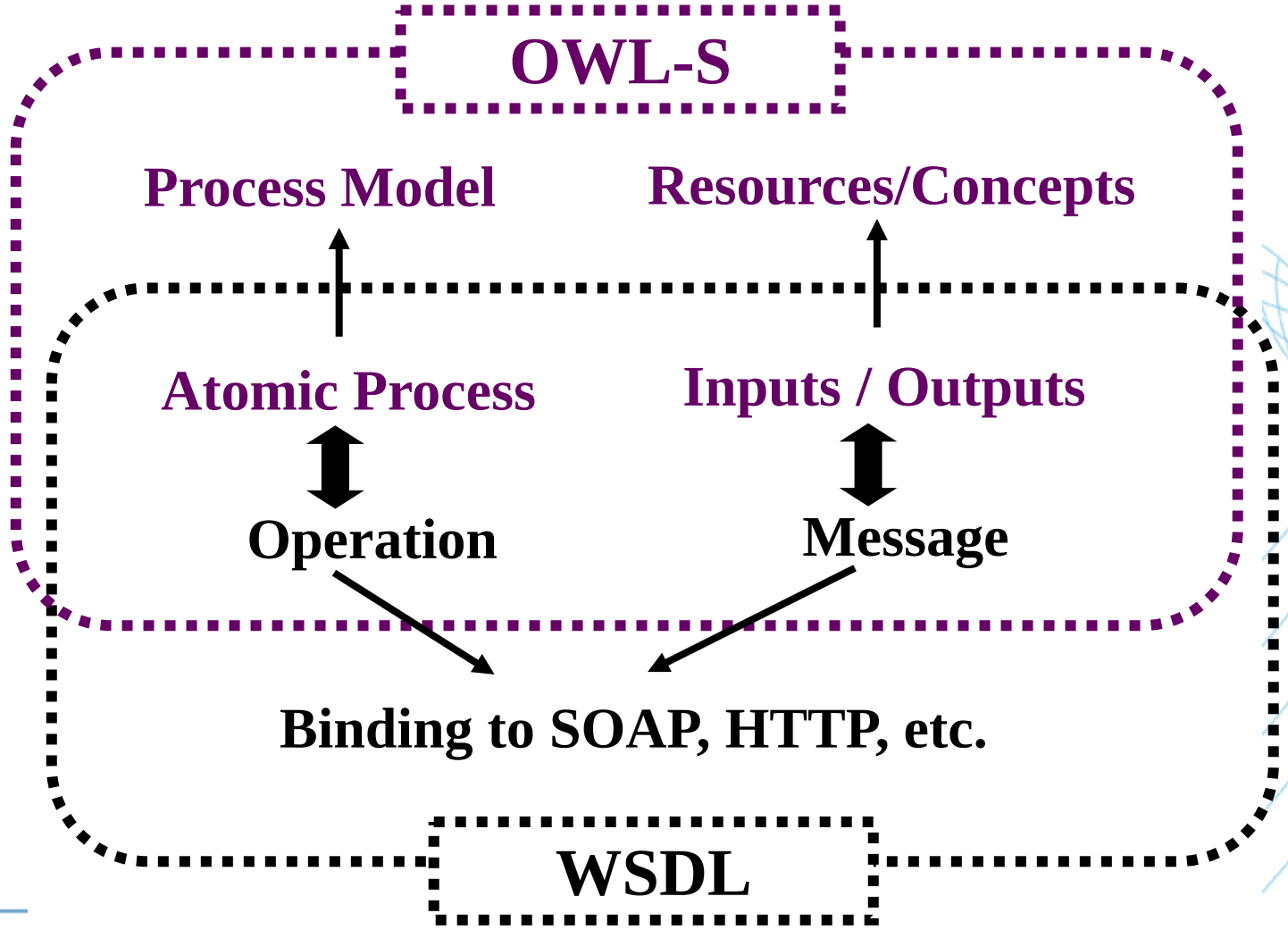


Service Grounding

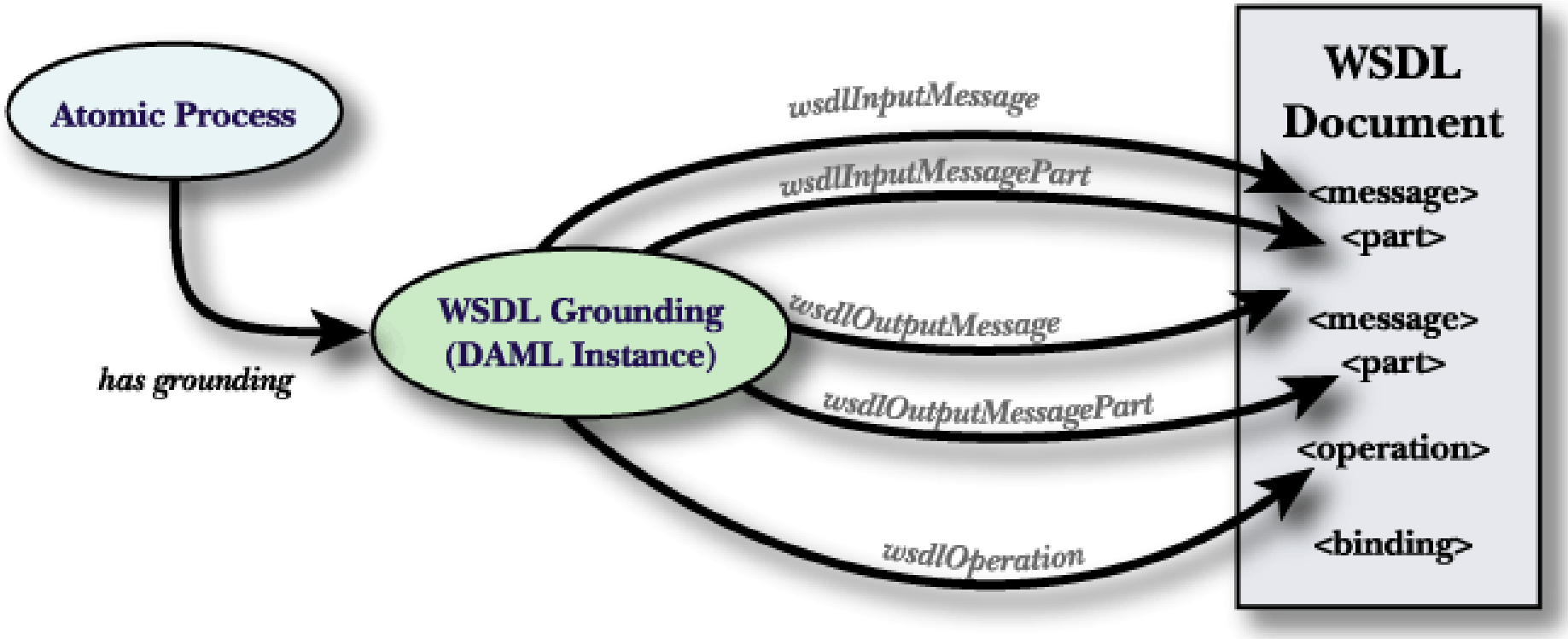


- Service Grounding
  - 提供服务访问信息的规范
  - **Service Model + Grounding** 才给定了使用服务的所有信息
  - 构建在**WSDL**之上
- 指定:
  - 通信协议、传输机制等

OWL-S / WSDL Binding



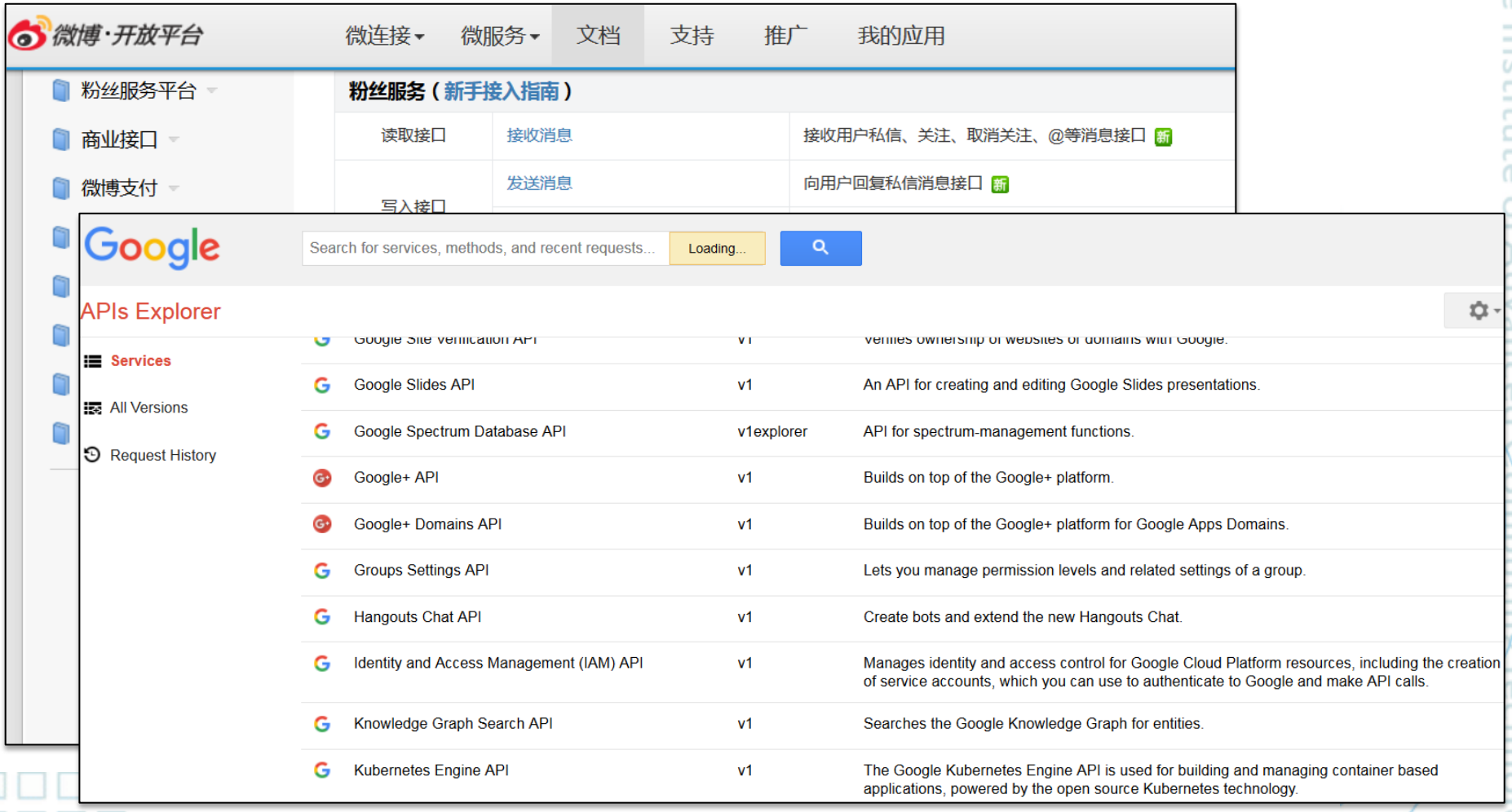
OWL-S / WSDL Mapping



OWL-S案例

- http://www.daml.org/services/owl-s/1.1/examples.html
- Congo.com (B2C site)

自然语言语义描述


微博·开放平台
微连接
微服务
文档
支持
推广
我的应用
粉丝服务平台
商业接口
微博支付
粉丝服务（新手接入指南）
读取接口
接收消息
接收用户私信、关注、取消关注、@等消息接口 新
写入接口
发送消息
向用户回复私信消息接口 新
Google
Search for services, methods, and recent requests...
Loading...
APIs Explorer
Services
All Versions
Request History
Google Slides API v1 An API for creating and editing Google Slides presentations.
Google Spectrum Database API v1explorer API for spectrum-management functions.
Google+ API v1 Builds on top of the Google+ platform.
Google+ Domains API v1 Builds on top of the Google+ platform for Google Apps Domains.
Groups Settings API v1 Lets you manage permission levels and related settings of a group.
Hangouts Chat API v1 Create bots and extend the new Hangouts Chat.
Identity and Access Management (IAM) API v1 Manages identity and access control for Google Cloud Platform resources, including the creation of service accounts, which you can use to authenticate to Google and make API calls.
Knowledge Graph Search API v1 Searches the Google Knowledge Graph for entities.
Kubernetes Engine API v1 The Google Kubernetes Engine API is used for building and managing container based applications, powered by the open source Kubernetes technology.

服务标注(Tag)

基于WSDL的文本分析模式

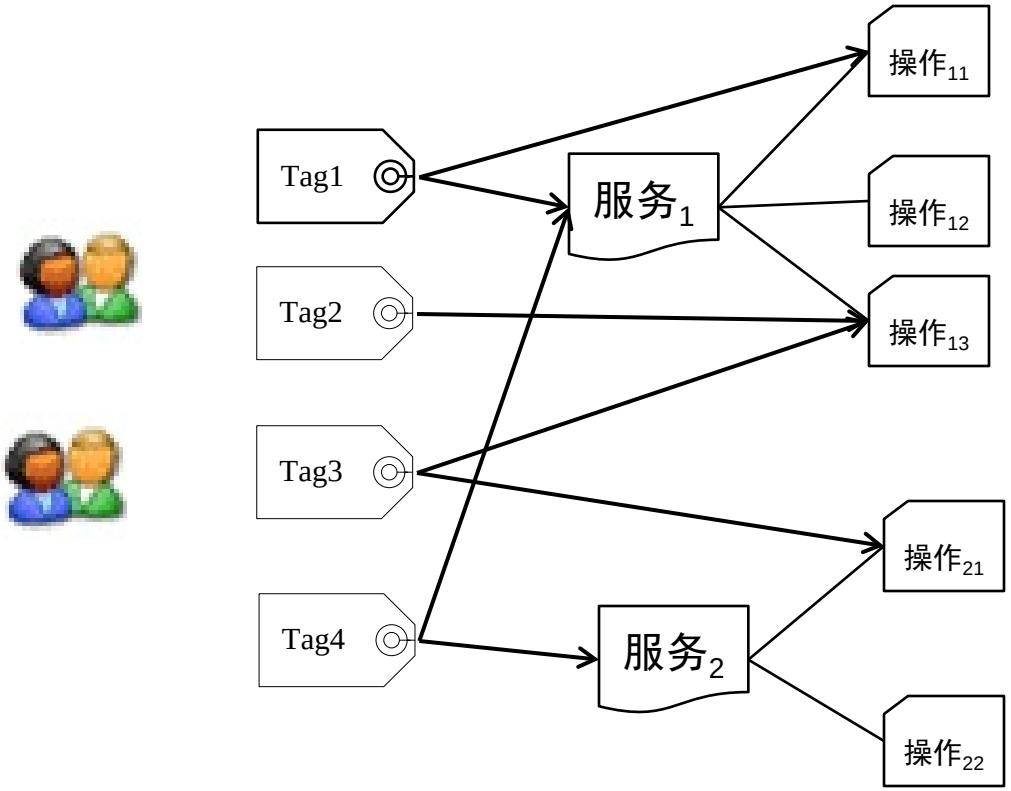
- 高效、易实现
- 准确率低、受WSDL本身限制

基于语义Web服务模式

- 准确率高
- 复杂、效率低、需要定义逻辑语义库

A Ranking Method for Social-Annotation-Based Service Discovery. The 6th IEEE International Symposium on Service-Oriented System Engineering (SOSE), 2011

服务标注模型



- 对Tag的挖掘-建立用户模型

不同用户描述习惯差异带来的问题

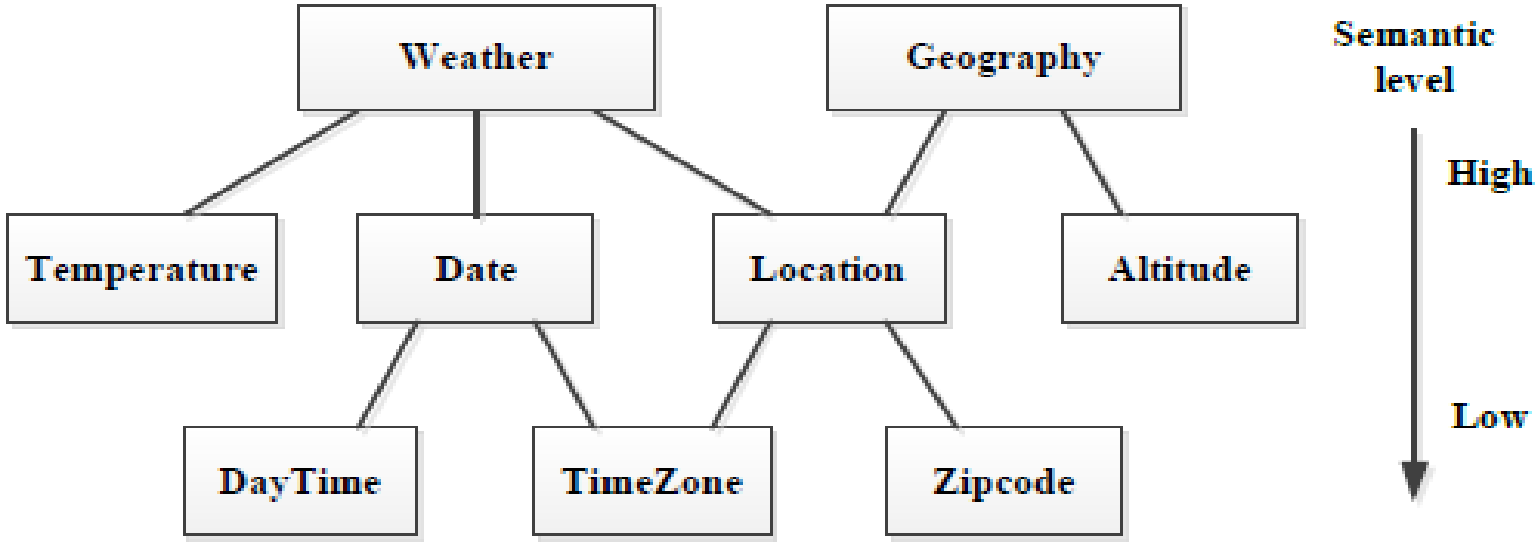
例如
“dev”&”develop”
“fun”&”funny”

建立用户词汇模型

分析用户词汇与全局词汇间关系

对Tag的挖掘-语义关系

- 包含关系、相似相关关系
- Weather包含Temperature的语义; 而TimeZone与Zipcode语义相似；Temperature与Date的语义则是相关关系

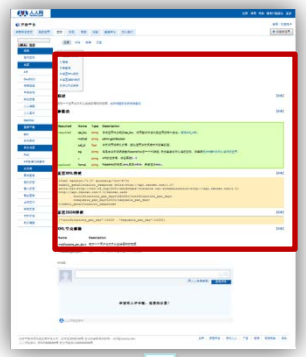


服务收集：以RESTful为例

网页主数据区的提取

RESTful Web服务API详情页的识别

结构化数据的抽取与转换

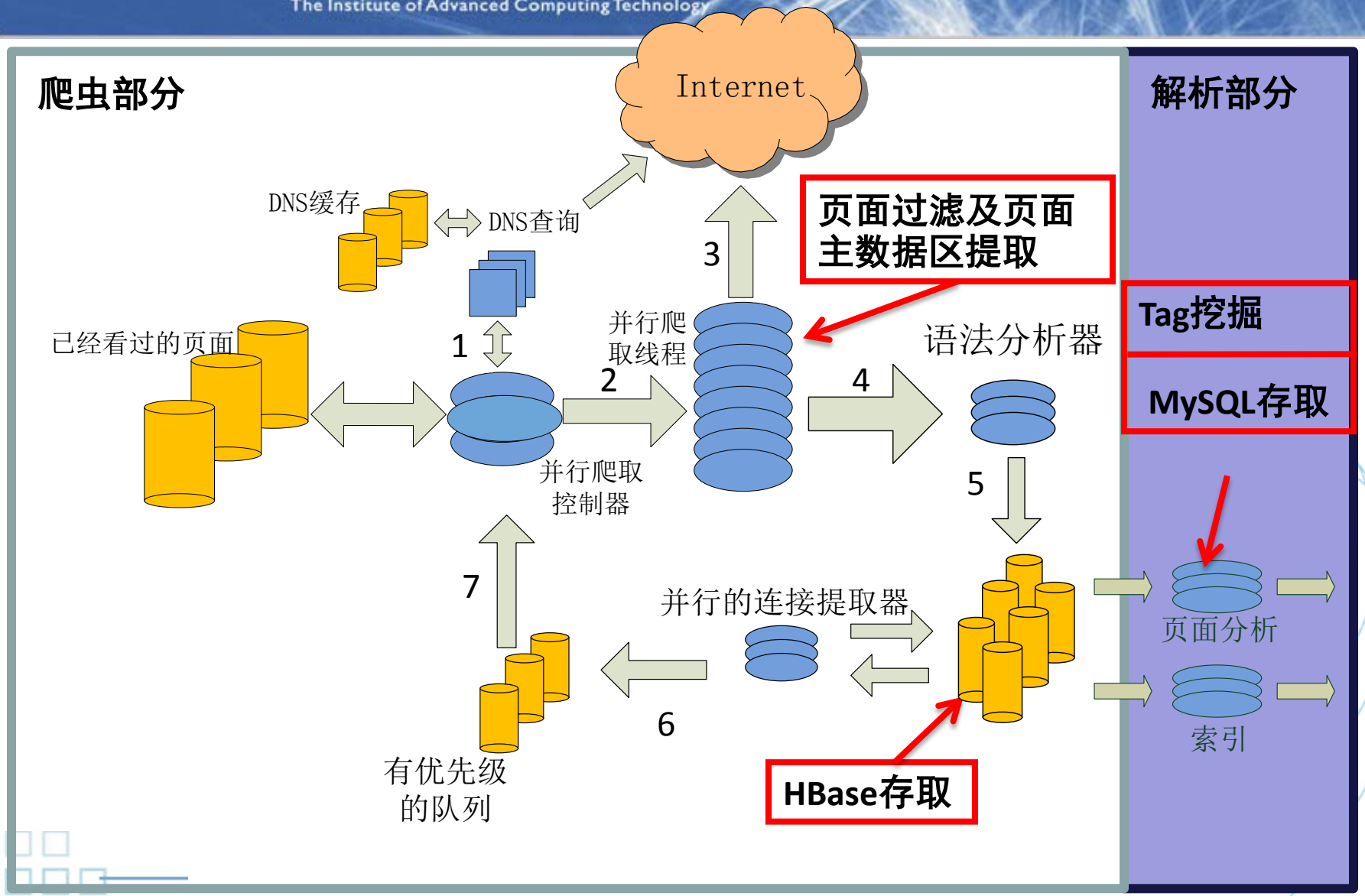


Y

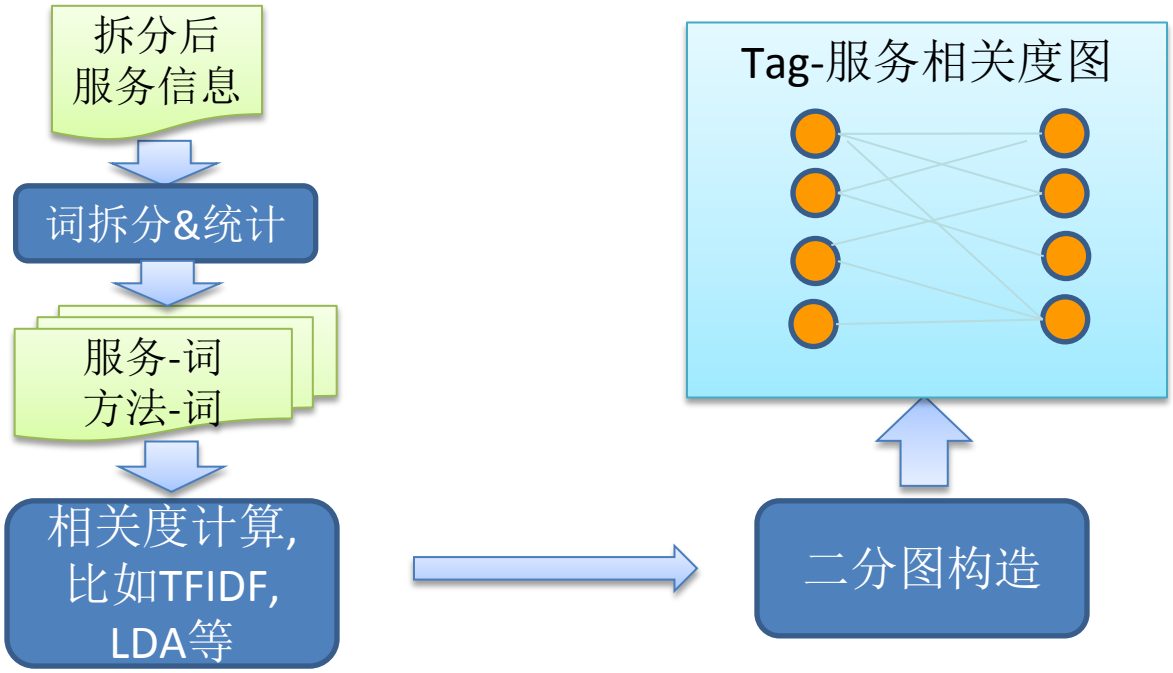
service表

operation表

message表


爬虫部分
解析部分
Internet
DNS缓存
DNS查询
3
页面过滤及页面
主数据区提取
Tag挖掘
MySQL存取
已经看过的页面
1
并行爬
取线程
2
4
语法分析器
并行爬取
控制器
5
7
并行的连接提取器
页面分析
6
索引
有优先级
的队列
HBase存取

服务的Tag挖掘

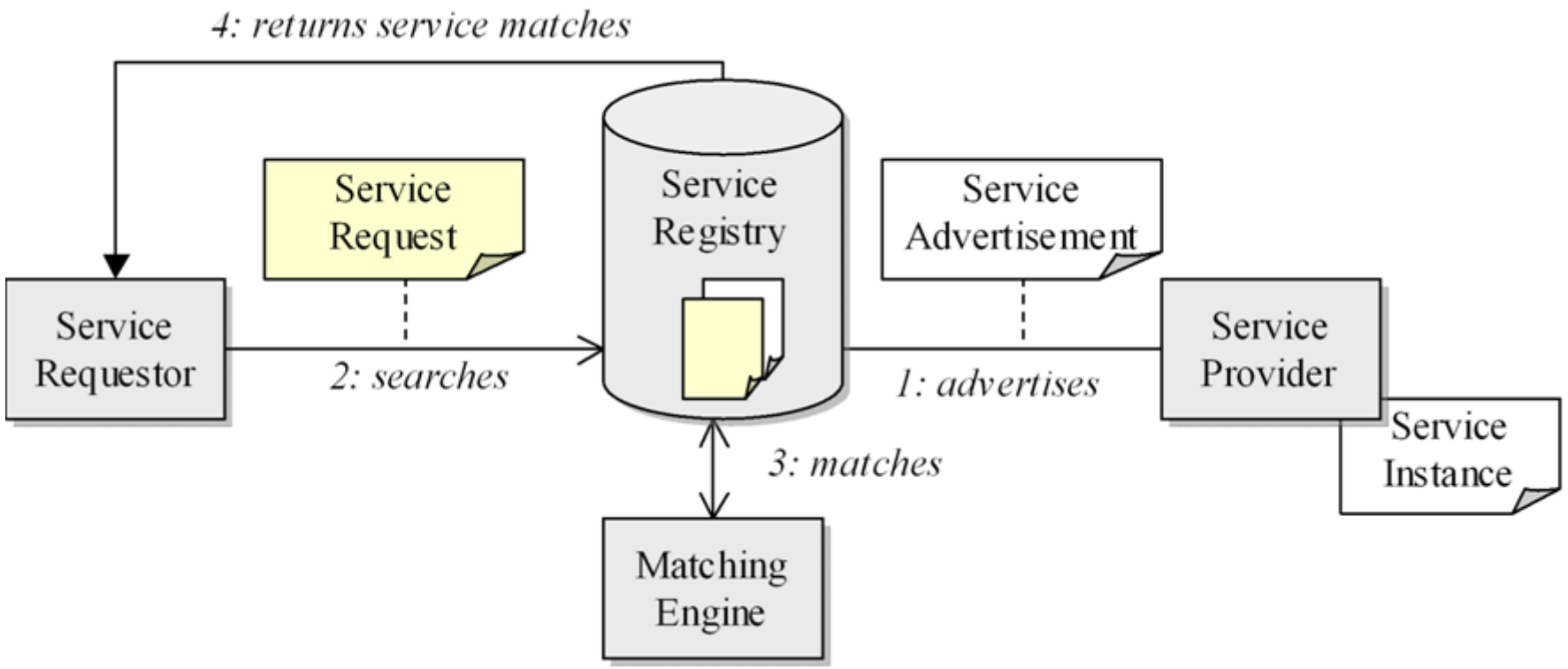


4.3 服务发现

服务发现

- 与其他Web资源一样，Web服务也需要定位和访问
  - 类似于网站与搜索引擎的关系
- Web服务被语义信息标注后，可基于其确切的能力进行服务发现，而不是简单地依据服务的接口
- 服务发现是一种根据Web服务语义信息来定位服务的方法

服务发现的参考架构



服务发现方法

- 语义Web服务发现
- 基于自然语言处理的服务发现
- 基于语法和结构的服务发现

架构元素

- Service Registry
  - 服务的“黄页”
- Matching Algorithm
  - 在匹配引擎中实现
  - 影响服务发现的效果
- Service Request
  - 捕获请求者的信息需求
- Service Advertisement
  - 描述服务
  - 由服务提供者创建

假设：相同的格式

Web服务描述

- WSDL
  - XML language for textual service description
- UDDI
  - Data model and API for service publication/searching
  - Contains links to WSDL documents
  - Main elements:
    - businessEntity, businessService, bindingTemplate, tModel

语义Web服务发现架构

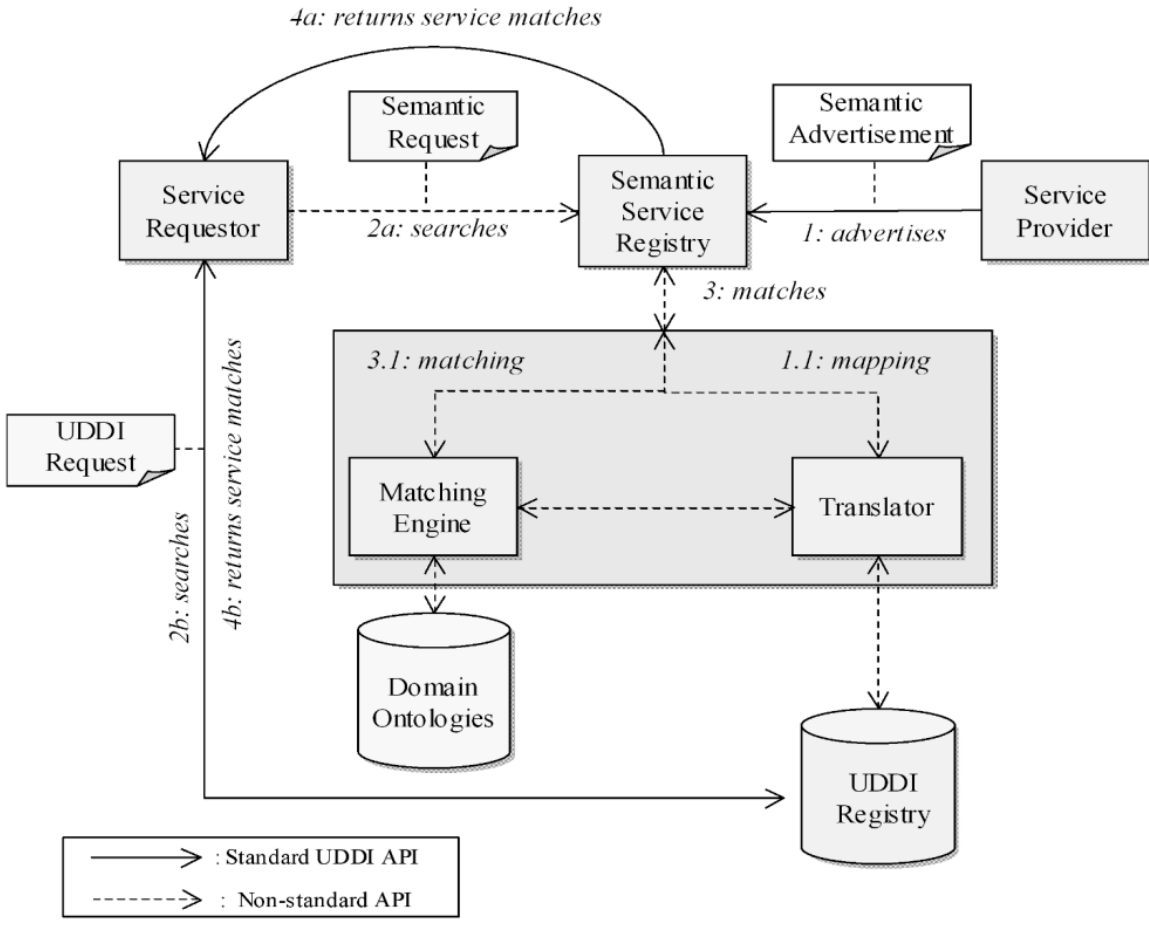
- **新的元素**
  - 服务标注本体**Service Annotation Ontologies (SAO)**
    - 形式化的服务描述模型
    - 给定服务能力
    - OWL-S, WSMO, WSDL-S, SWSO
  - 领域本体
    - 特定领域的词汇
    - 替换服务描述中的关键字和文本
    - 概念和关系的层次
    - 用OWL, DAML+OIL, RDF(S), …描述

参考架构中基本组件的扩展

- Service Registry
  - 在UDDI中增加语义描述的引用
- Matching Algorithm
  - 更复杂、智能
  - 充分利用服务描述的形式语义
- Service Advertisement
  - 由SAO描述
  - 引用领域本体的概念
- Service Request
  - 通常和advertisement相似
  - 本体集成和语义调解，可用于桥接不同的请求-广告规范

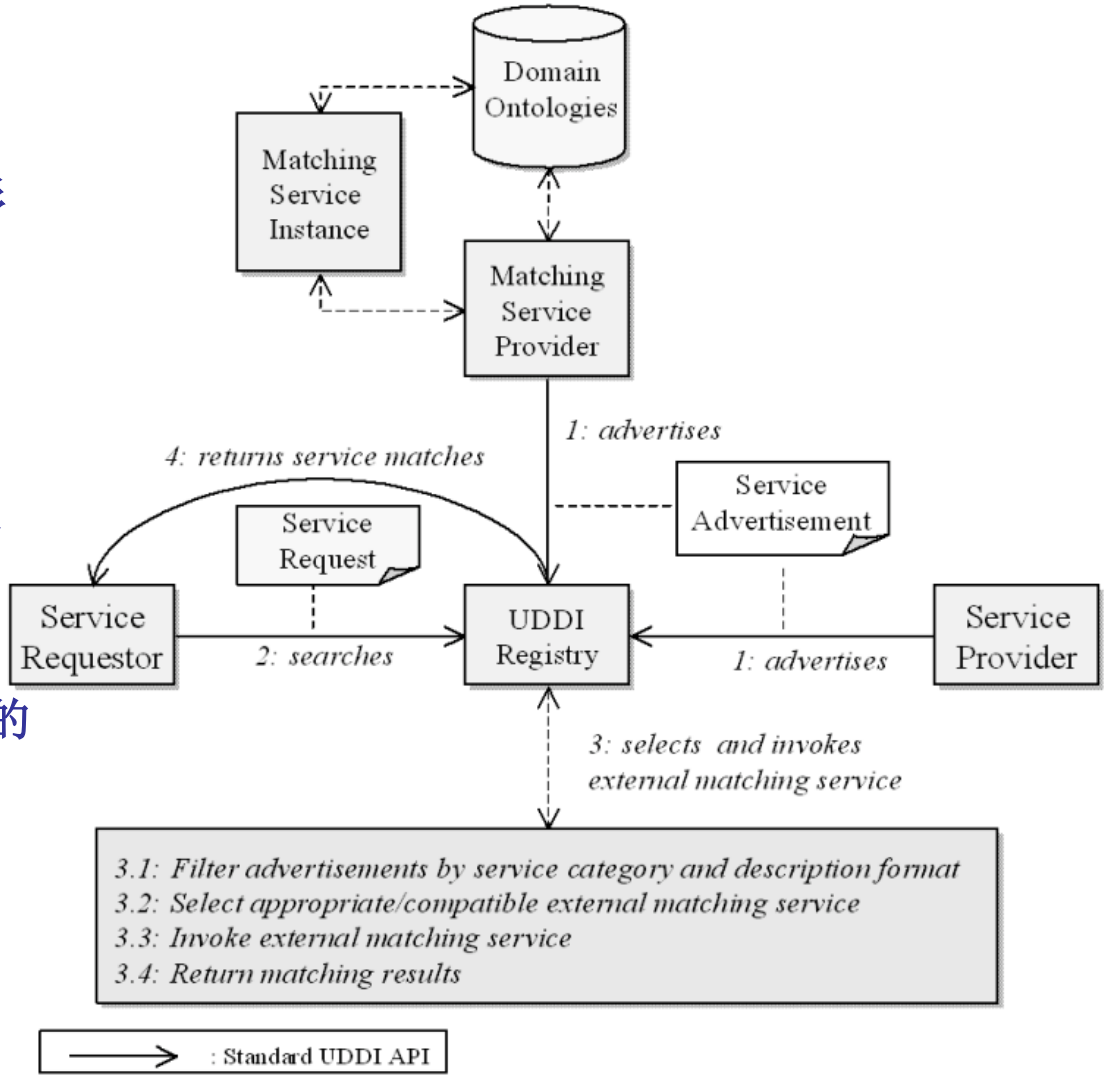
中心化架构

- 架构1：Importing Semantics to UDDI
  - 对UDDI的语义扩展
  - tModels用于语义描述
  - Translator创建语义tModels
  - Semantic matching在一个外部引擎中执行
  - 也支持基于关键字的匹配
  - 需UDDI查询API进行一定的扩展



中心化架构

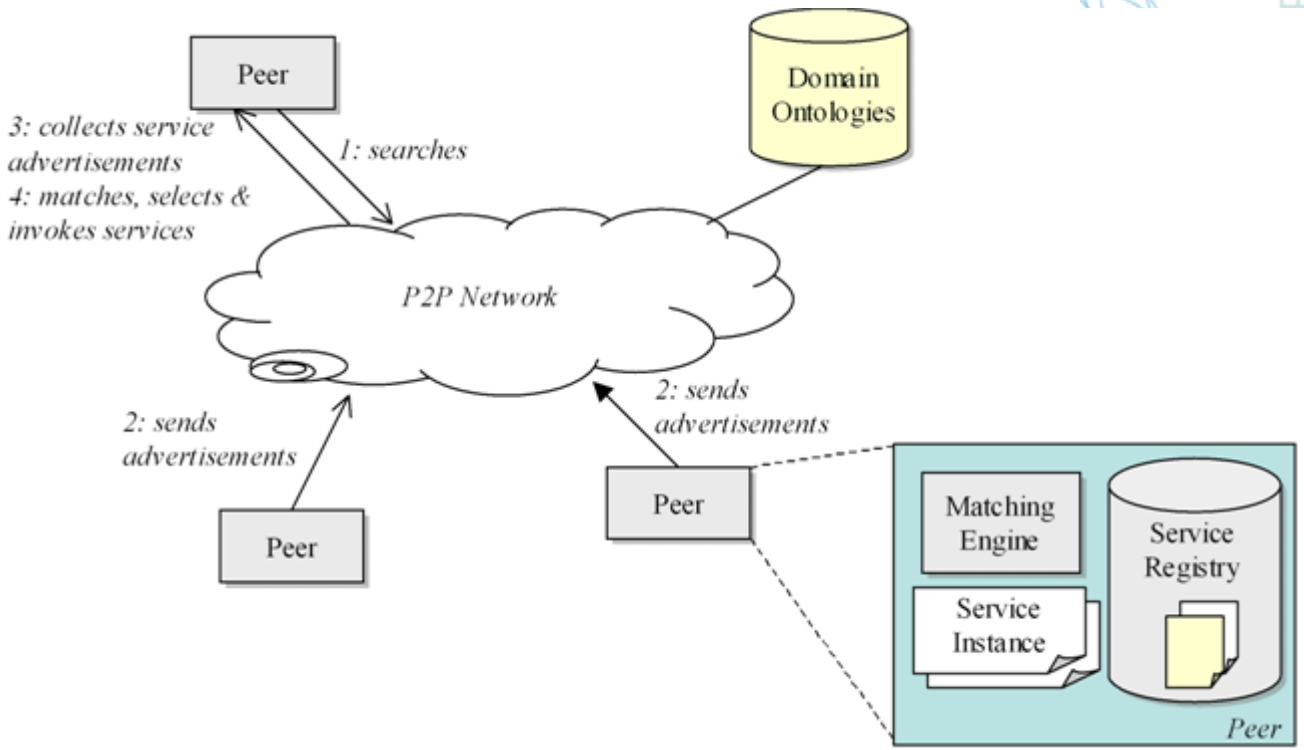
- 架构2：External Matching
  - Services匹配算法以Web服务形式提供
  - 支持多样化的SAO和匹配算法
  - Step1：Ad hoc地选择最佳匹配服务
  - Step2：以请求作参数调用选择的服务
  - 对UDDI API的修改尽可能小
  - 运行更灵活的业务模型，但服务组合更复杂



P2P架构

- P2P适合于(可扩展、高效的)分布式环境，Peer是服务请求者或提供者
- 每个peer请求者可使用自身的匹配算法
- 每个peer提供者可直接更新本地服务广告

Result: high flexibility



语义Web服务发现的方法/算法

- service profile
  - during discovery, the requestor is more interested in whether a service can fulfill the posed requirements than in how it works or how it can be invoked
  - IOPE attributes, described in the service profile
- 匹配度Degree of Match (DoM)
  - *表达两个实体(services, IOPE attributes or specific service operation等)相似程度的一个值*
  - *语义Web服务匹配方法的重要特征*
  - *运行对发现服务进行排名*
  - *举例：* exact, plugin, subsumes, subsumed-by, fail

不同的匹配方法

- 直接
  - 返回匹配请求的单个服务
- 间接
  - 计算服务组合
- 基于逻辑
  - Description Logics and First Order Logic 推理

语义能力匹配

- 主要思想
  - *一个服务**A** 匹配一个请求 **R**，当**R**的所有输出与**A**的所有输出匹配，**R**的所有输入与**A**的所有输入匹配*
- 输入输出间使用描述逻辑的包容subsumption 匹配
- Outputs 认为比inputs更重要

Paolucci, M., Kawamura, T., Payne, T.R., & Sycara, K.P. (2002a). Semantic matching of Web services capabilities. In I. Horrocks & J. Hendler (Eds.), First International Semantic Web Conference on the Semantic Web, Sardinia, (LNCS 2342, pp. 333-347). Springer-Verlag.

匹配类型

Subsumption relation	Meaning/Potential problems
req.i subsumes adv.i	More specific input information might be required for the proper execution of the service described by *adv*
adv.i subsumes req.i	The request contains all input information required for the proper execution of the service described by *adv*
req.o subsumes adv.o	The output is valid for the requestor, though it may contain only a fraction of the desired results. The requestor may need a set or composition of such services in order to fulfill its requirements
adv.o subsumes req.o	The service may not be able to produce the required outputs. In extreme cases, the service execution results may be totally irrelevant to the request. However, we usually make the assumption that the service provider provides more specific services, too

Degree of Match	Matching conditions
EXACT	**If req.o is *equivalent(等价)* to adv.o, or If req.o is a direct subclass of adv.o**
PLUGIN	**If adv.o *subsumes(包含于)* req.o**
SUBSUMES	**If req.o *subsumes(包含于)* adv.o**
FAIL	**If there is no subsumption relationship between req.o and adv.o**

匹配算法

Main algorithm: match(req, A)

1: matchedServices = {}
2: **For all** adv in A **do**
3: **If** ((DoMI=Imatch(req, adv))!=FAIL **AND** (DoMO = O match(req, adv)) !=FAIL)
4: adv.DoM=min (DoMI, DoMO)
5: matchedServices=matchedServicesÈ{adv}
6: **Return** sortByDoM(matchedServices)

Function: Omatch (req, adv)

7: DoM={}
8: **For all** req.o in req.O **do**
9: **For all** adv.o in adv.O **do**
10: DoM=DoMÈ{degreeOfMatch(req.o, adv.o}
11: **Return** min(DoM)

Function: Imatch(req, adv)

12: DoM={}
13: **For all** adv.i in adv.I do
14: **For all** req.i in req.I do
15: DoM=DoMÈ{degreeOfMatch(adv.i, req.i)}
16: **Return** min(DoM)

多层匹配

- 方法1的一种变体
- 主要思想
  - 考虑功能和非功能性的服务数据
- 多层匹配
  - IOPE(input, output, precondition, and effects)属性、服务类别、定制的服务参数(如QoS相关)
- 匹配度(DoM)聚合
  - 衡量不同层次的DoM，赋予不同的权值
  - 非常难的优化问题

Main algorithm: match(req, adv)

```
1: DoMI=allImatch(req.I, adv.I)
2: DoMO=allO match(req.O, adv.O)
3: DoMSC=SC match(req.SC, adv.SC)
4: DoMP=PAR match(req.par, adv.par)
5: If (DoMI == FAIL OR DoMO == FAIL OR DoMSC == FAIL OR DoMP == FAIL)
6:    Return FAIL
7: Else
8:    Return (DoM = aggregate (DoMI, DoMO, DoMSC, DoMP))
```

描述逻辑匹配

- 方法1基于subsumption matching
- 通过Service Profile Ontology进行matching
  - 服务由DL Concept Expression表示，包括所有服务约束（inputs, ourputs等）
  - DL推理器创建本体树
  - 一个基于逻辑的服务注册中心
- DL subsumption matching
- 重新定义了方法1中的DoM集合
- 引入新的DoM[Li, 2004]
  - 一个advertisement匹配一个请求，当他们的交集可满足

Li, L., & Horrocks, I. (2004). A software framework for matchmaking based on Semantic Web technology. International Journal of electronic Commerce, 6(4), 39-60.

举例-约会服务

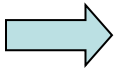
FreeDatingService≡DatingService⊓∃hasServiceProfile.9ServiceProfile
⊓∃requiresPayment.FreeOfCharge)

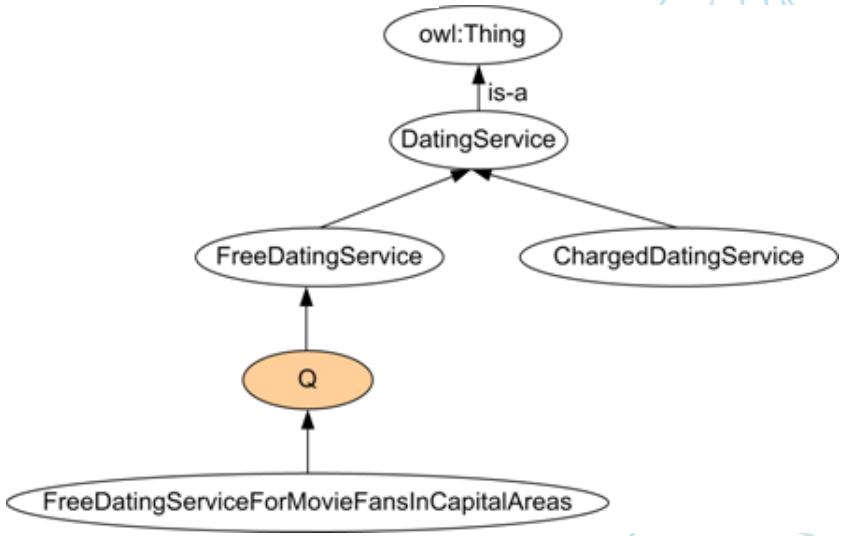
FreeDatingSErviceForMovieFansInCapitalAreas≡DatingService
⊓∃hasServiceProfile.(ServiceProfile⊓∃requiresPayment.FreeOfCharge
⊓∃hasPersonalProfile.(ContactProfile⊓∃hasLocation.CapitalArea)
⊓∃has InterestProfile.(InterestProfile⊓∃hasInterest.Movies))

Q≡DatingService⊓∃hasServiceProfile (ServiceProfile⊓∃requiresPaymentFreeOfCharge
⊓∃hasPersonalProfile (ContactProfile⊓∃hasLocation.UrbanArea)
⊓∃hasInterstProfile (InterestProfile⊓∃hasInterest.Entertainment))

2 Advertisements and a Request Q

The Service Profile Ontology after DL reasoning

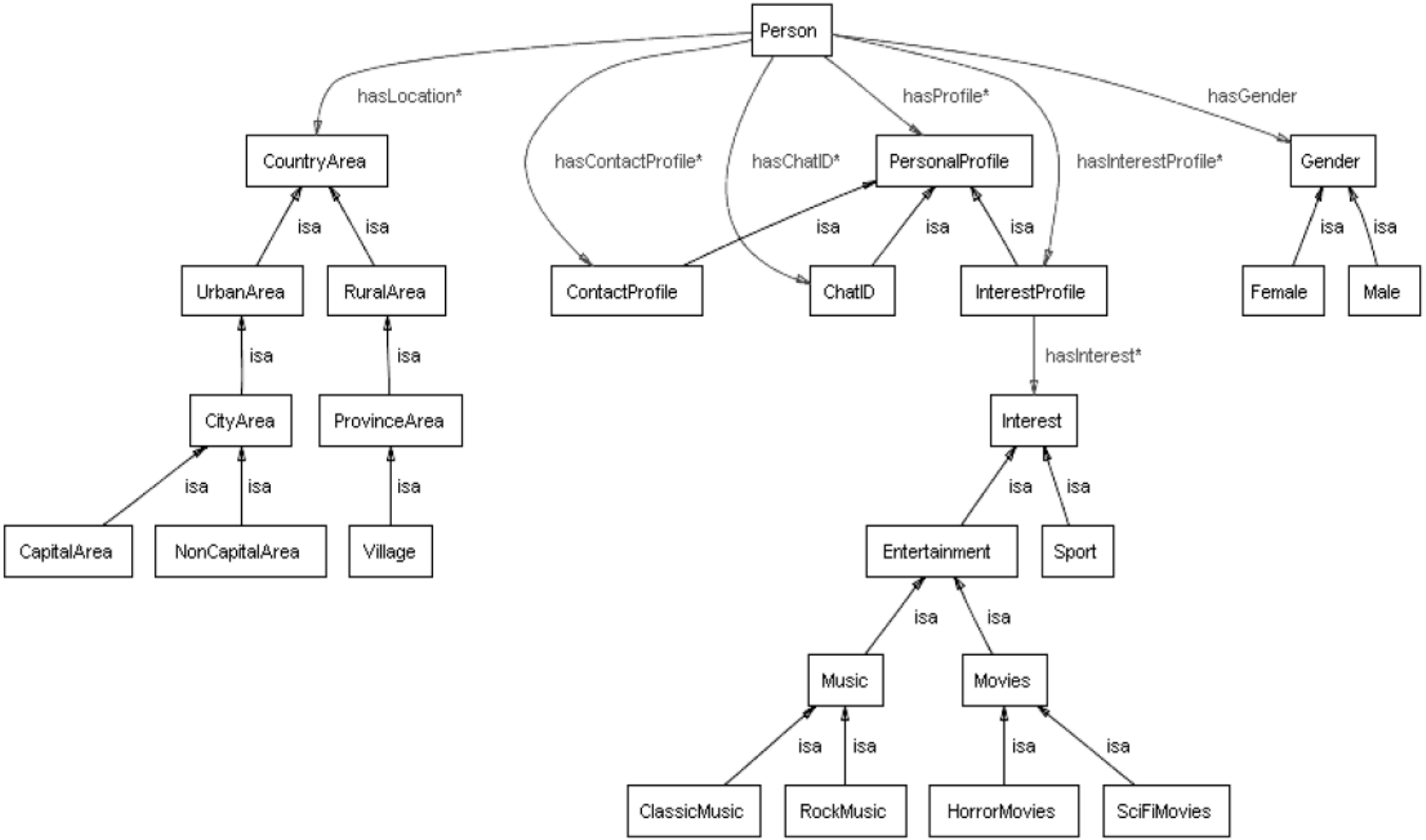




DoM(Q,FreeDatingService) = ***PLUGIN***
DoM(Q,FreeDatingServiceForMovie…) = ***SUBSUME***

**Assumption: PLUGIN is better than SUBSUME*

service profile ontology



服务发现方法

- 语义Web服务发现
- 基于自然语言处理的服务发现
- 基于语法和结构的服务发现

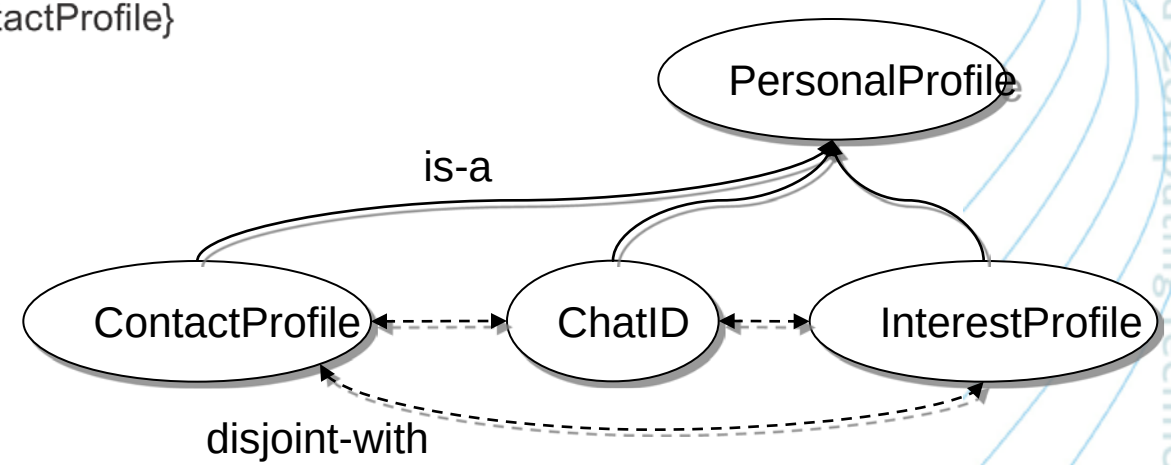
逻辑匹配的局限

- **纯逻辑的匹配方法会产生有悖常理的结果，如：**
  - **R:**
    - input: InterestProfile ⊓ ∃hasInterest.SciFiMovies
    - output: ContactProfile
  - **A :两个服务，只有第二个会匹配**

```
Find_Interests_Of_Female_MSN_Us-
    ers:  input={Person⊓∃hasGender.Male}
    output={InterestProfile, ChatID}
Find_Person_by_Interest:  input={InterestProfile}
    output={ContactProfile}
```

DoM(R,A) = FAIL

Reason: output of R is ***disjoint with*** output of A although their inputs are "logically relevant"



相似性度量

- **相似度度量方案 – 主要思想**
  - 运行评价服务相似性更灵活的方法
  - **FAIL, LOGICAL FAIL, SIM_{IR}.**

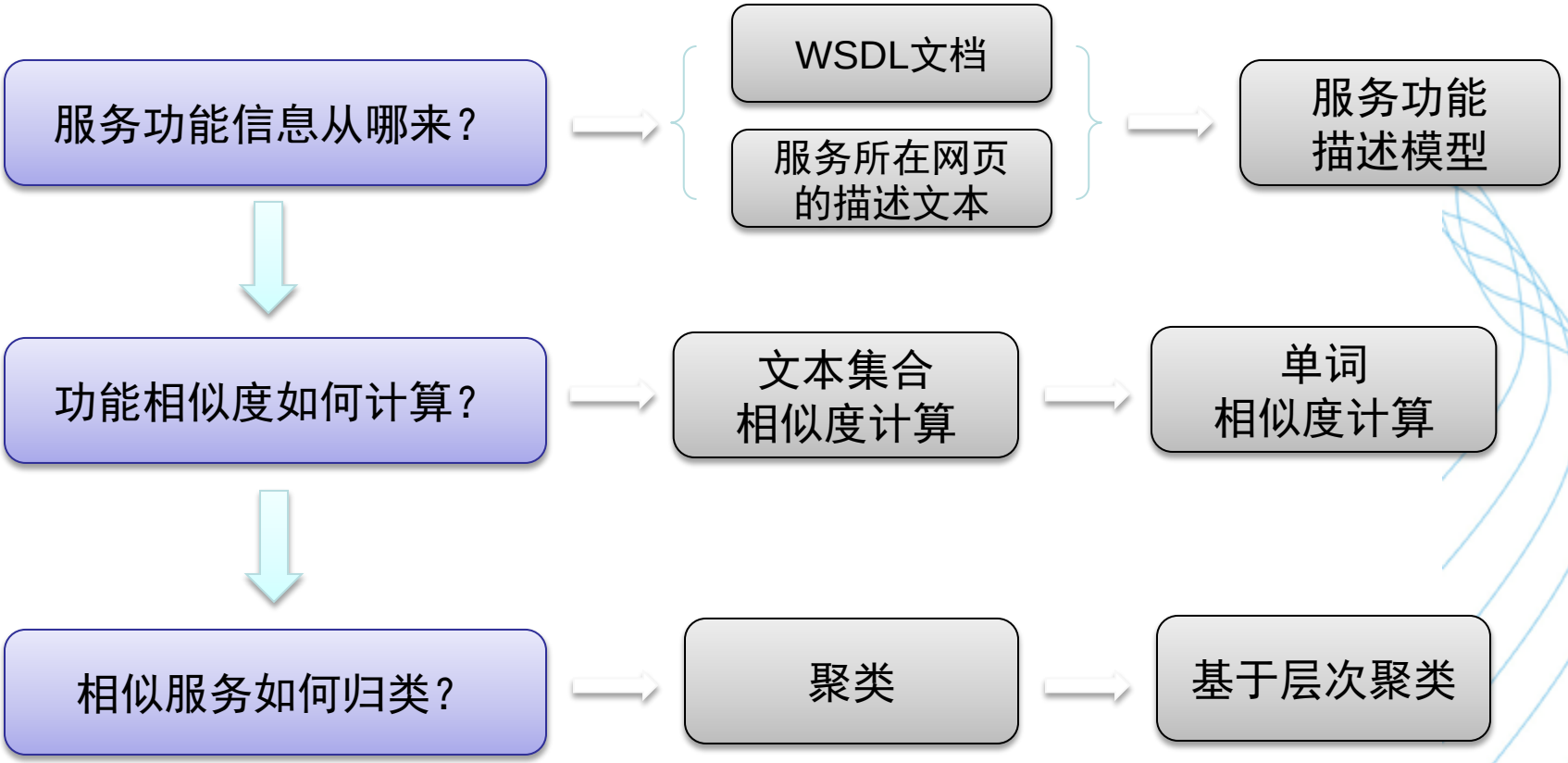
Main algorithm: match(req, adv)
1: DoM=logicalMatch(req, adv)
2: **If** DoM == LOGICAL-FAIL
3: **If** sim(req, adv) threshold
4: **Return** FAIL
5: **Else**
6: **Return** SIM_{IR}
7: **Else**
8: **Return** DoM

Function: logicalMatch(req, adv)
9: **Return** min(Omatch(req, adv), Imatch(req, adv))

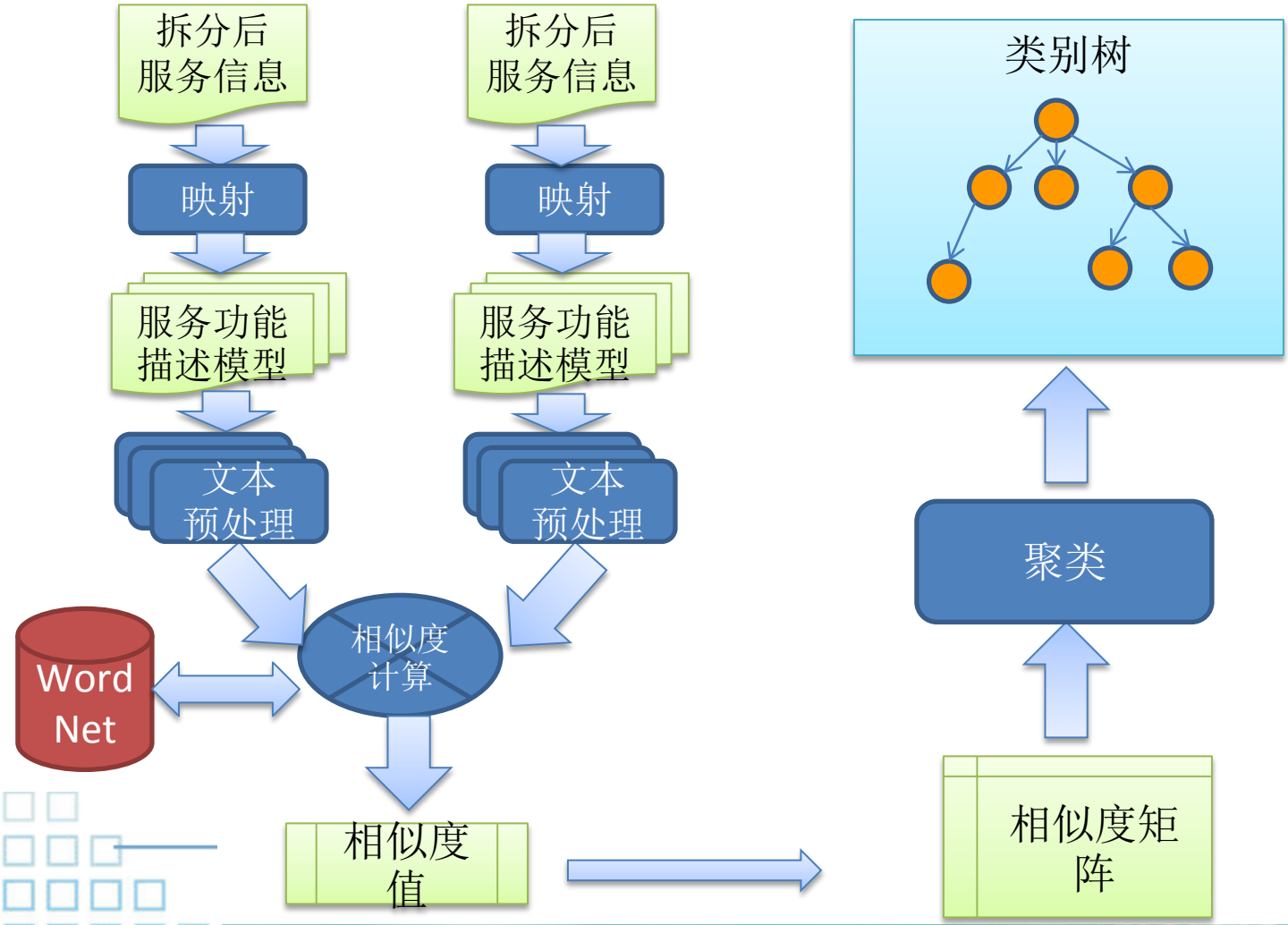
Function sim(req, adv)
10: *Any similarity measure*

- **基于自然语言处理的相似性度量**
  - 逻辑只是“相关”的一个方面
  - 核心是度量服务、方法、输入输出等自然语言文本的相似性

服务的功能相似性挖掘


服务功能信息从哪来？
WSDL文档
服务所在网页的描述文本
服务功能描述模型
功能相似度如何计算？
文本集合相似度计算
单词相似度计算
相似服务如何归类？
聚类
基于层次聚类

功能相似关系挖掘流程



服务功能描述模型

WSDL文档

```
<wsdl>
  ..
  <Message>…</Message>
  <Message>…</Message>
  ..
  ..
  <portType>
        <operation>…</operation>
        ..
  </portType>
  ..
  <Service>
        ..
  </Service>
  ..
```

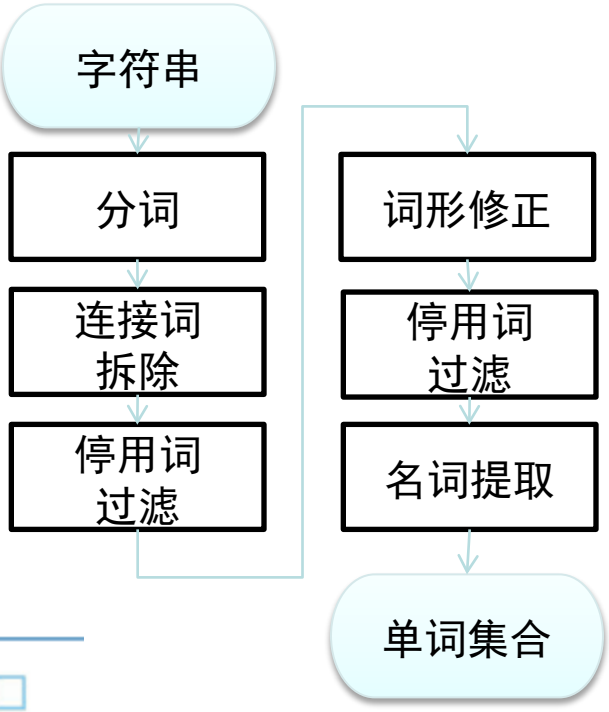
服务功能描述模型

- 标题信息T
- 主体信息B

服务功能描述模型的文本信息中有很多无用文本或非标准文本
如何处理？

文本预处理

- 文本预处理动机
  - **原始本文信息粗糙，不准确**
  - **提高相似度计算的效率和精确度**
- 预处理流程



服务功能相似度计算

- 服务的相似度计算

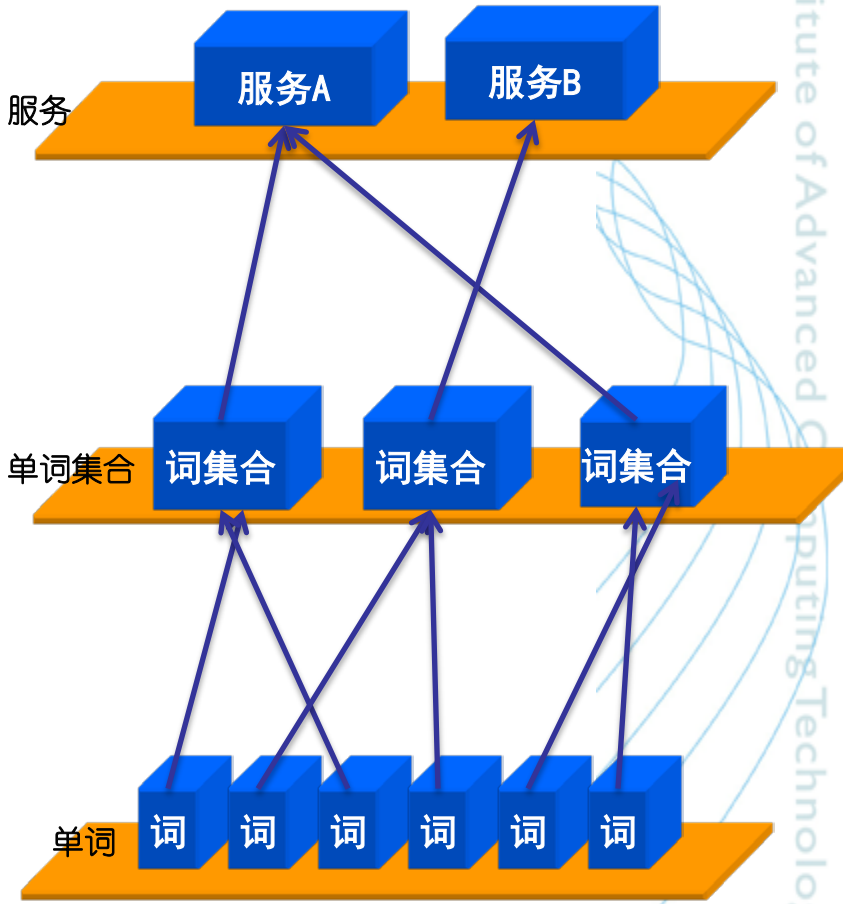
$$SimService(W_1, W_2) = \alpha * SimSet(W_1.T, W_2.T) + \beta * SimSet(W_1.B, W_2.B) + \gamma * SimSet(W_1.A, W_2.A)$$

- 关键词集合的相似度计算

$$SimSet_{ps}(S_1, S_2) = \frac{\sum_{w \in S_1} Sim_m(w, S_2) + \sum_{w \in S_2} Sim_m(w, S_1)}{|S_1| + |S_2|}$$

- 单词的相似度计算

$$Sim(w_1, w_2) = -\log \frac{\left[\min_{c_1 \in sen(w_1), c_2 \in sen(w_2)} len(c_1, c_2)\right]}{2d_{\max}}$$

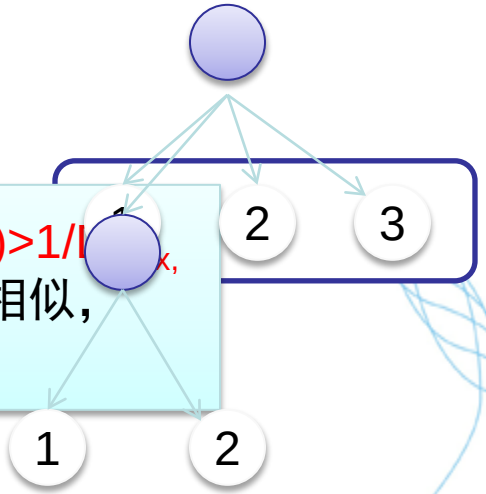


聚类

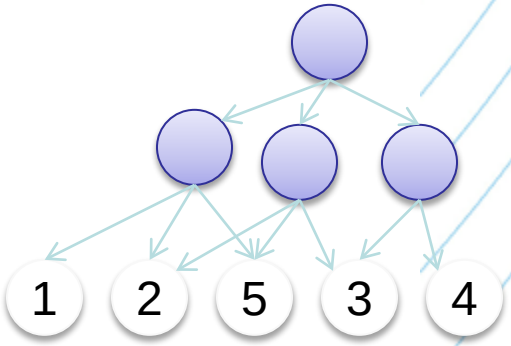
- 自主递增式聚类算法
  - **自主决定类别数目**
  - **支持增量式聚类**
  - **考虑分裂与合并**
- 算法参数
  - **允许最大类直径(D_{max})**
- 子算法
  - **最近/次近类别查找算法**
  - **类别分裂算法**
  - **类别合并算法**
  - **类别更新算法**

若Sim(服务A，服务B)>1/D_{max}，
则认为两服务功能相似，
可被聚为一类。

类别分裂示意流程



类别合并示意流程



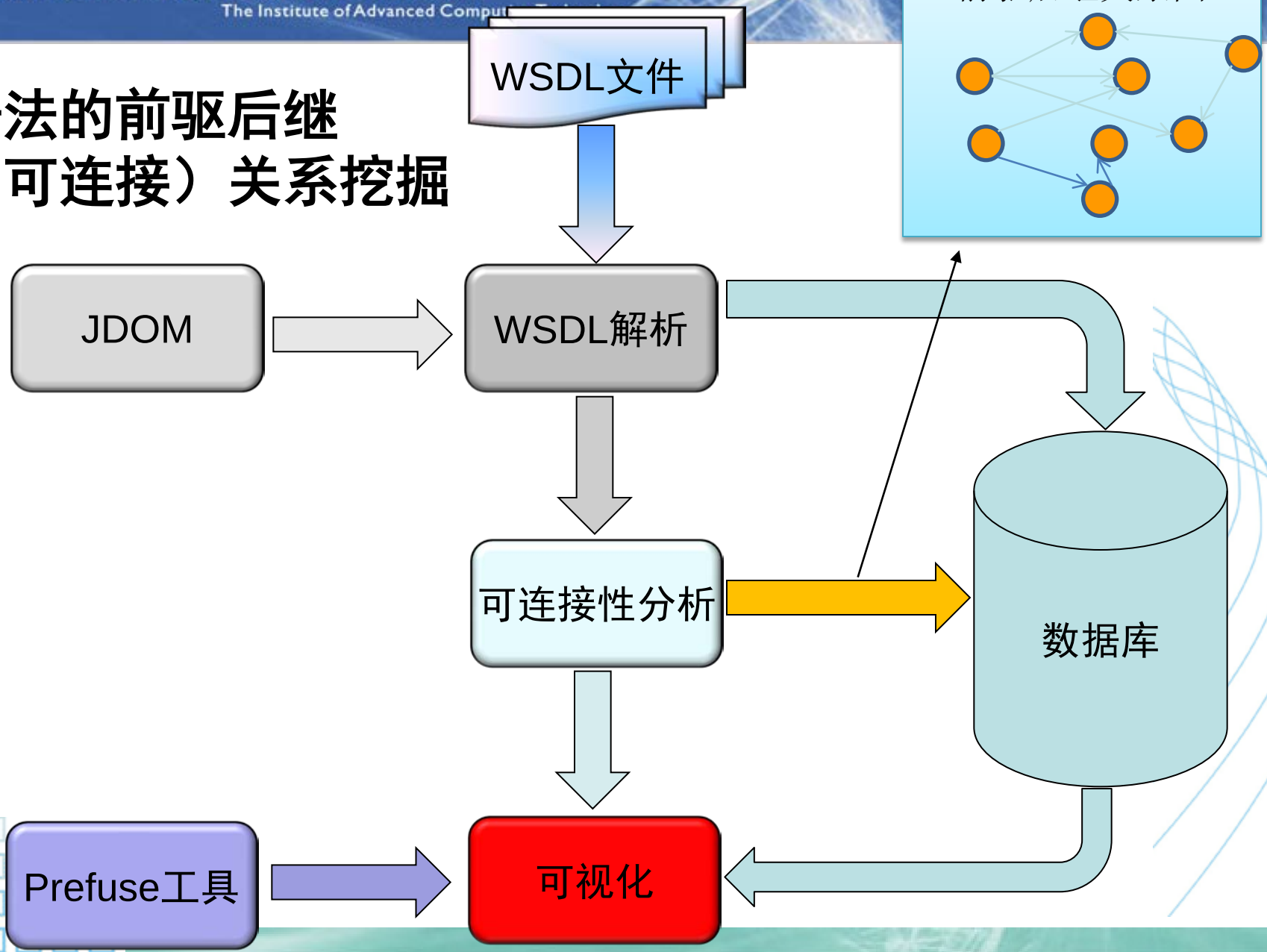
自然语言理解技术

- 传统方法
  - 词频统计：TFIDF等
  - 机器学习：LDA, LSA等
  - 浅语义：WordNet
- 神经网络
  - Word2Vec, Document2Vec
  - TextCNN
  - RNN
  - Seq2Seq

服务发现方法

- 语义Web服务发现
- 基于自然语言处理的服务发现
- 基于语法和结构的服务发现

语法的前驱后继（可连接）关系挖掘



可连接模式分析

- 输入输出的语法结构
  - –输出参数可以通过调换顺序、数据类型兼容、有选择传输等达到匹配输入参数的目的
- 参数顺序
  - ◆当参数按照顺序进行比较的时候只要按顺序从队列中提取参数即可
  - ◆当参数没有顺序时，只要统计参数的个数，比较两个方法的参数类型的个数就可以确定它们之间的关系
- 参数半匹配、全匹配分析：
  - ◆当全匹配时要求两个方法的参数个数完全一致。
  - ◆半匹配时输出参数的个数大于或等于输入参数的个数，系统可以根据输入参数将一些不合适的输出参数给剔除掉

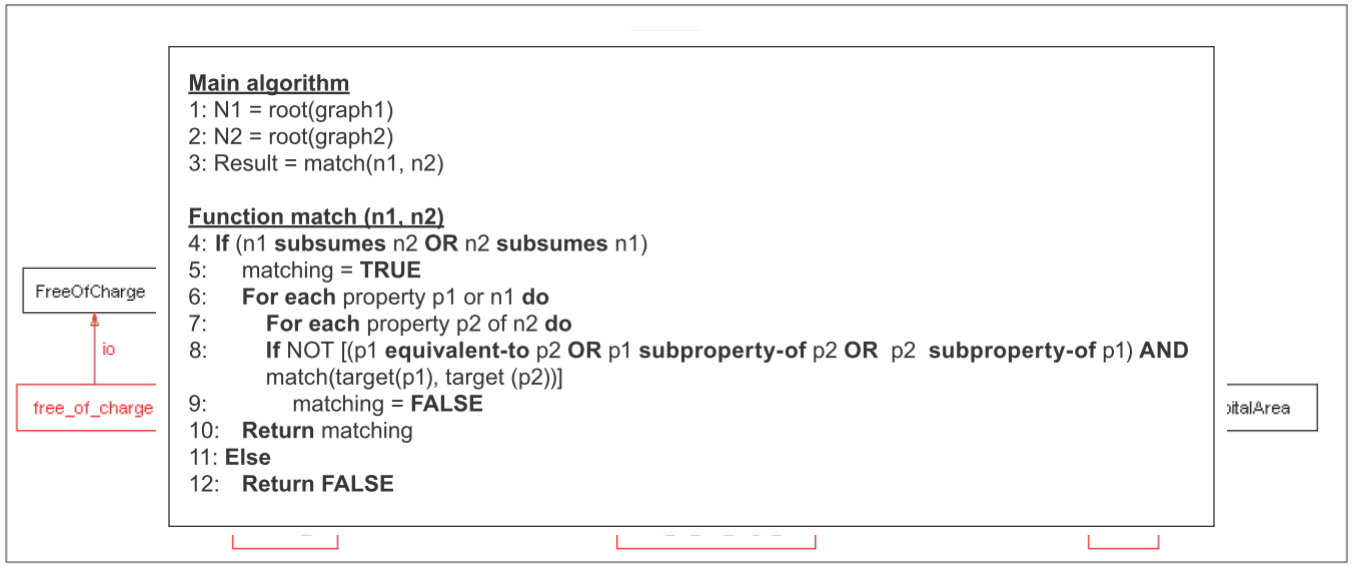
可连接模式分析

- 根据上面的分析产生了八种可连接模式：
  - 顺序、全匹配、类型一致模式
  - 顺序、全匹配、类型兼容模式
  - 顺序、半匹配、类型一致模式
  - 顺序、半匹配、类型兼容模式
  - 乱序、全匹配、类型一致模式
  - 乱序、全匹配、类型兼容模式
  - 乱序、半匹配、类型一致模式
  - 乱序、半匹配、类型兼容模式

基于图的匹配

- 服务被表示成DAG
  - Nodes ~ individuals of concepts
  - Arcs ~ roles between individuals
- 主要思想
  - 结构匹配：两个服务描述匹配，当它们有相同的结构，并且相应的结点匹配
- 已有图匹配算法可用

Figure 11. Graph representing the service S (io:instance-of, dashed line:concept role)



间接的基于图匹配

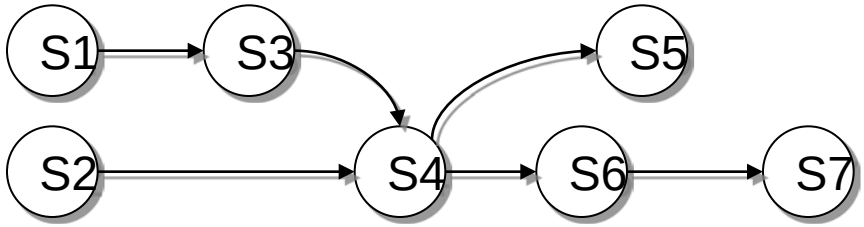
- 间接匹配
  - 复杂工作流组合
  - 最简单情况下的“服务链”
  - 服务链创建规则
    - 服务链中每个服务的输入，要么匹配请求的输入，要么匹配上一个服务的输出
    - 服务链最后一个服务的一个输出，要匹配请求的每个输出

举例

1: Service specifications

Service	Inputs	Outputs
S1	A, B	E
S2	A, B, C	F, N
S3	E, C	F
S4	F	K, M
S5	K, D	Z, Y
S6	K	D, Z
S7	D	Y

2: Service graph



3:

Request inputs:{A,B,C,D}
Request outputs:{Z,Y}

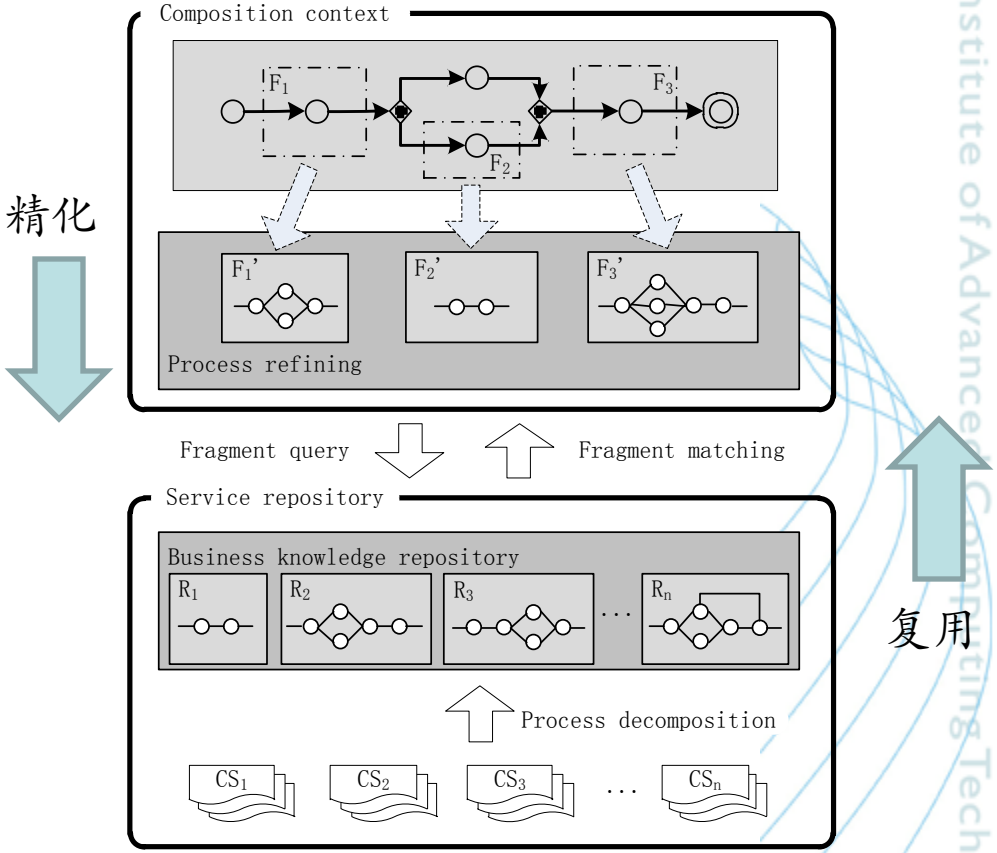
Discovered Service Chains

S1, S3, S4, S6, S7
S1, S3, S4, S5
S2, S4, S6, S7
S2, S4, S5

Policy-based service chain selection can be applied (e.g., the shortest)

重用已有组合服务的结构

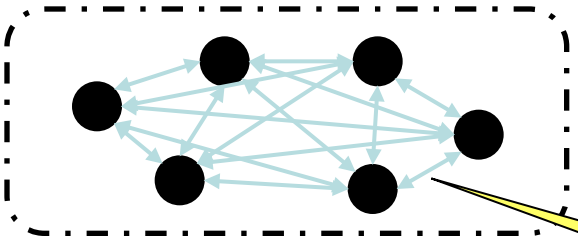
- 提高服务发现的粒度
  - **现有方法：组件服务**
  - **模块化、规范化的业务流程片段具有更高的参考价值**
- 借鉴程序挖掘的思想，抽取可复用组合流程片段



Business Process Decomposition based on Service Relevance Mining. WI-IAT, vol. 1, pp.573-580, 2010 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, 2010

片段可复用性的标准

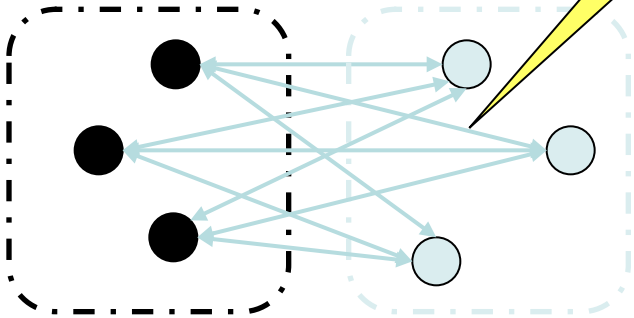
- 功能单一性：片段内部聚合度(Cohesion)越大越好



聚合度评估

服务的关联性

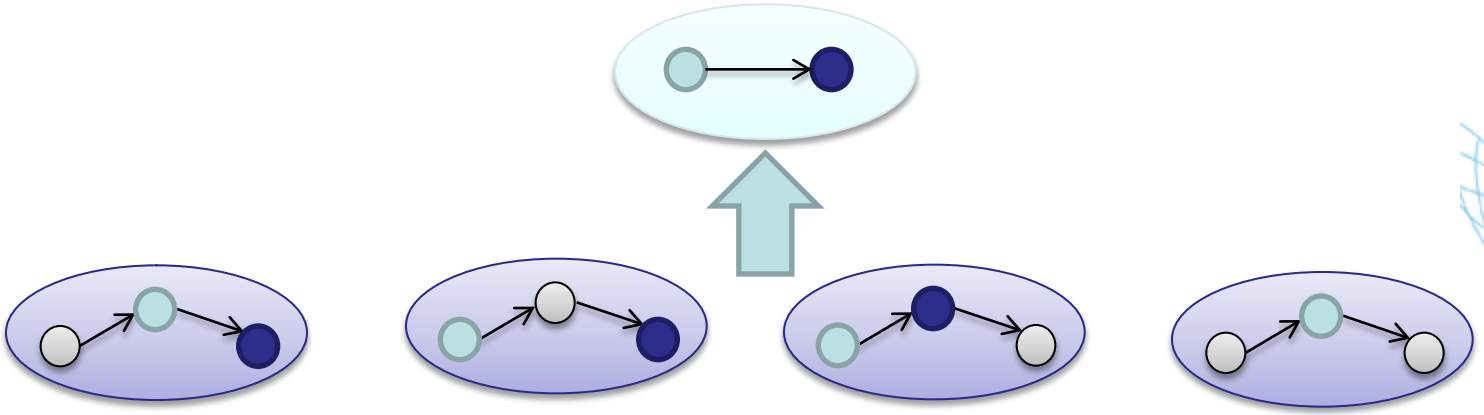
- 调用依赖：片段之间耦合度(Correlation)越小越好



耦合度评估

服务关联性评估

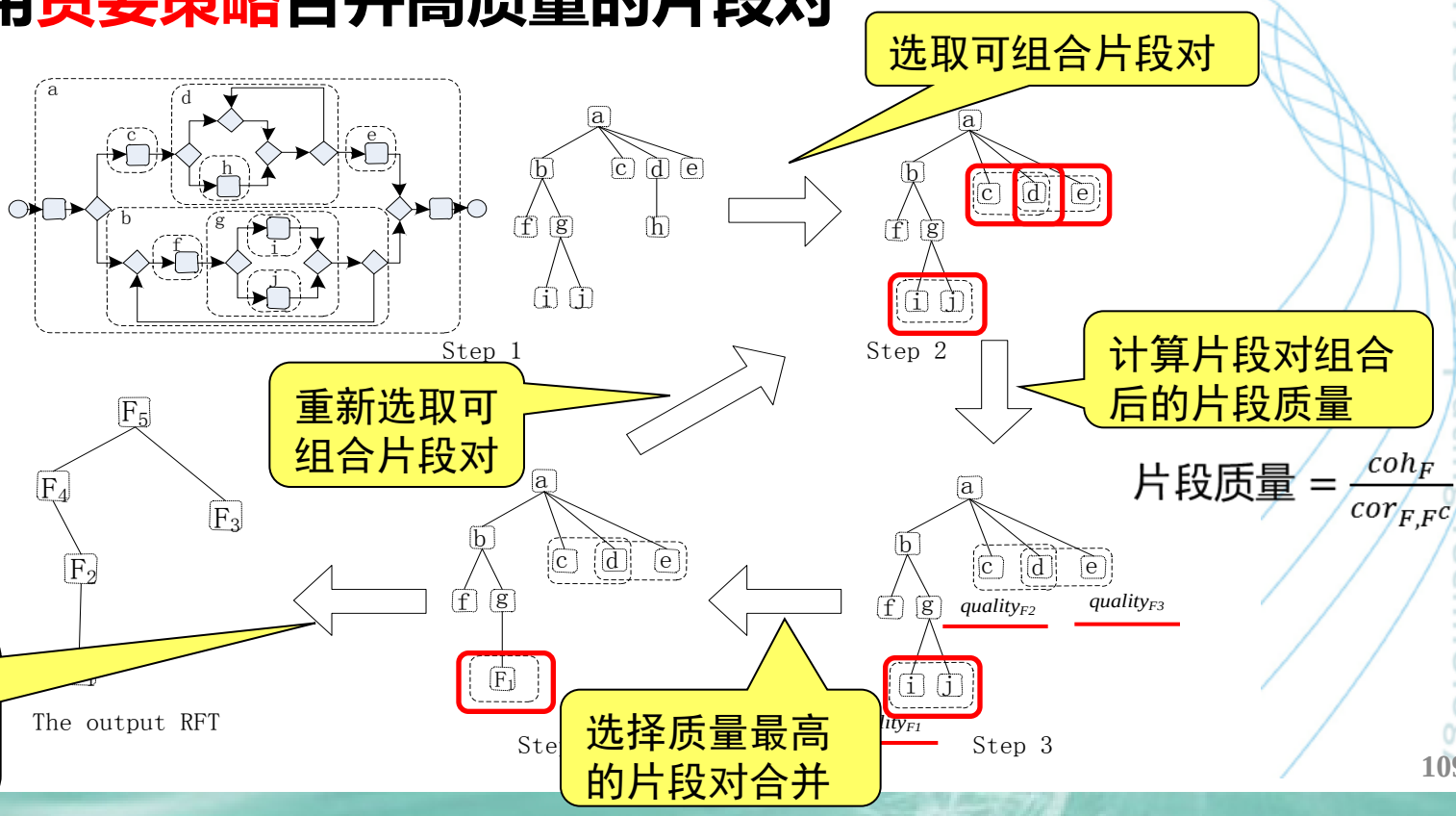
- 基本假设：服务频繁同现 → 领域关联性较强



- 关联规则：$s_i \rightarrow s_j\ (p, c)$，（服务关联程度的表示）
  - 置信度(c)：在服务s_i出现的流程中，服务s_j出现的概率，**服务关联程度衡量**
  - 支持度(p)：服务s_i 和s_j在所有流程中同时出现的概率，**规则的可信程度衡量**

流程分解基本思路

- (1)对流程进行分解，形成层次化的良构片段树
- (2)类似聚类分析，对片段树进行迭代收缩分析
  - 使用**贪婪策略**合并高质量的片段对



结 束!