Service Oriented Development/Engineering

Service Oriented (*Everything!*):

- Architecture
- Analysis and Design
- Requirements
- Modelling
- Software Engineering
- Business Logic/Process

Web services:

- Remote Procedure Call
- Middleware
- (Simple Object Access Protocol)
- Web Service Description Language
- Service Discovery
- Representational state transfer
- Semantic Web
- Ontologies and service taxonomies

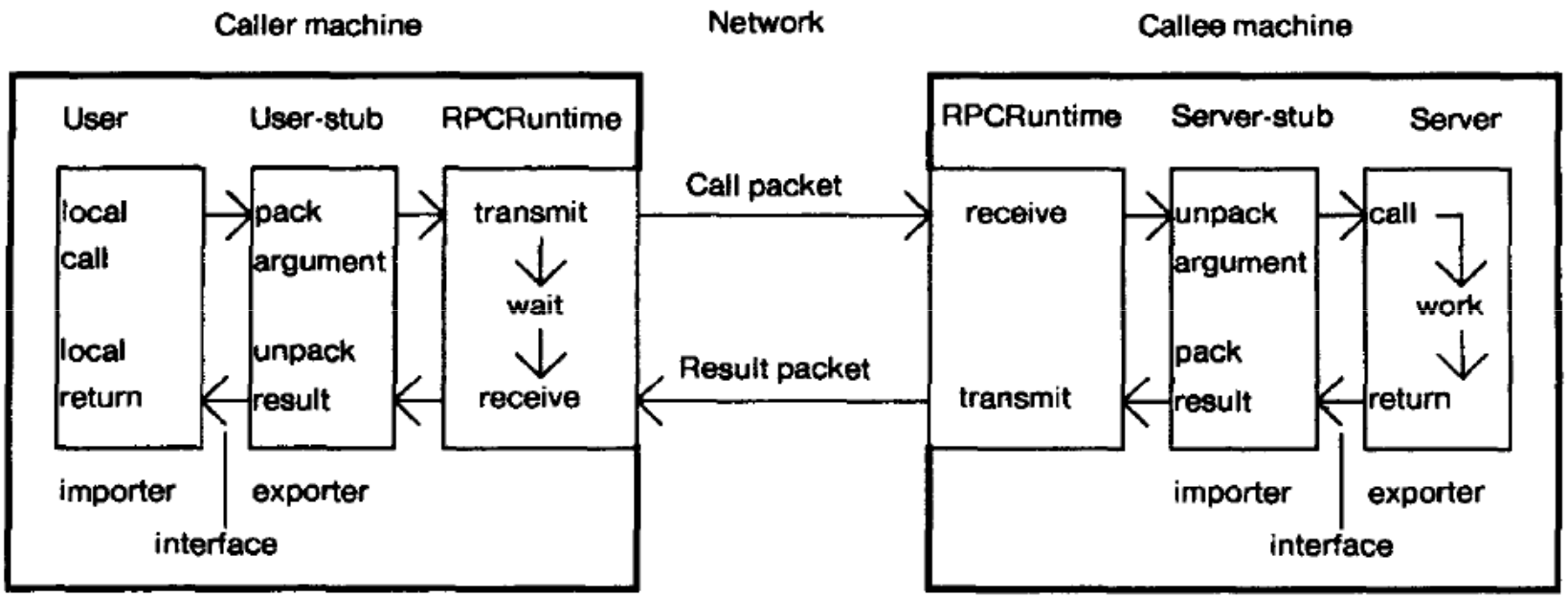
Services are not:

- Components
- Features
- Objects/Classes
- Procedures/Functions

Are they?

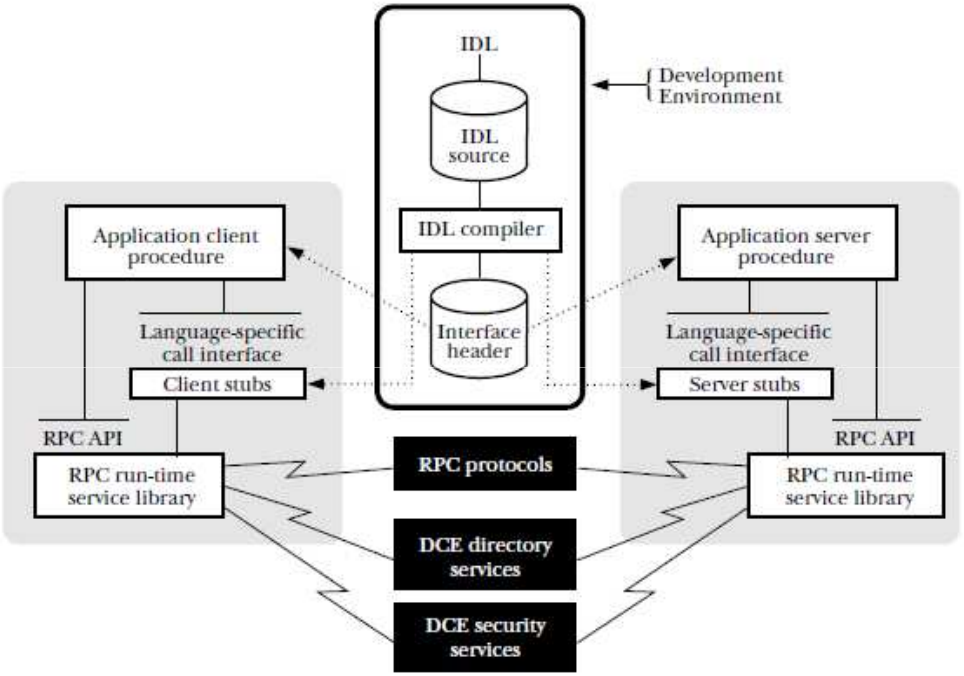
Implementing Remote Procedure Calls

Andrew D. Birrell and Bruce Jay Nelson,
ACM Trans. Comput. Syst. 2, 1
(February 1984), 39-59

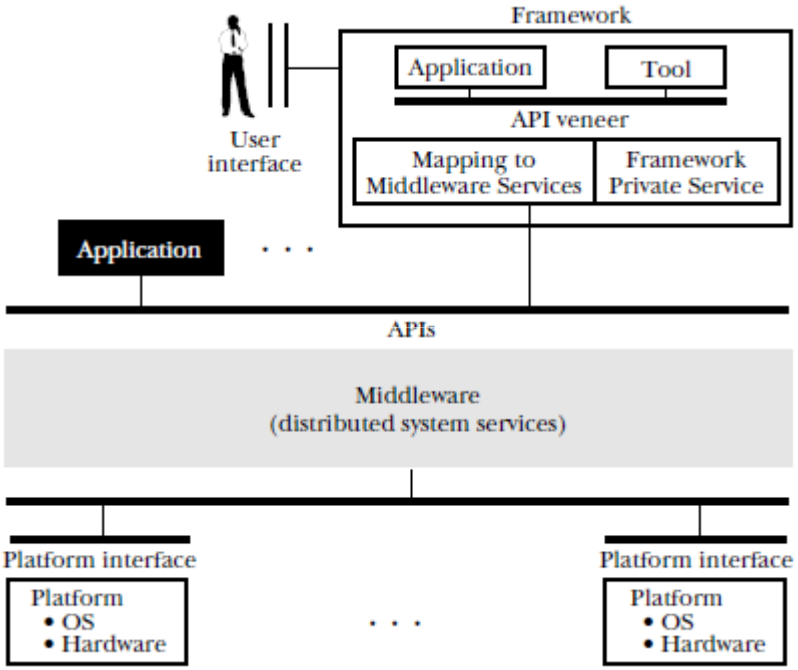


Middleware: a model for distributed system services

Philip A. Bernstein, *Commun. ACM* 39, 2 (February 1996), 86-98.



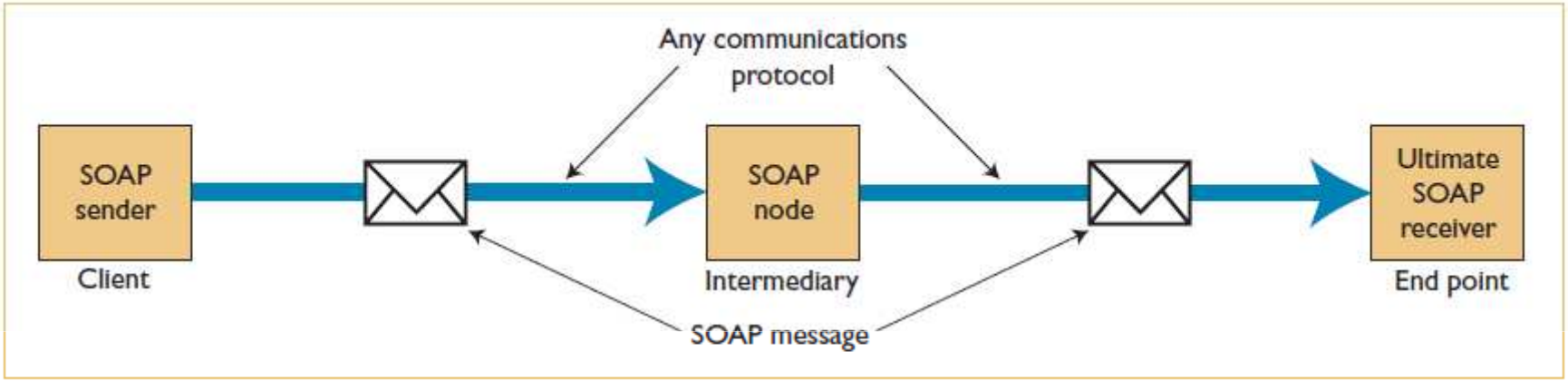
RPC Architecture



Middleware Architecture

Web Services Are Not Distributed Objects

Werner Vogels, *IEEE Internet Computing* 7, 6 (November 2003), 59-66



SOAP's transport independence:

Web service documents encapsulated in a SOAP message can be delivered directly to a destination over a single transport or via a collection of intermediaries over a variety of transports

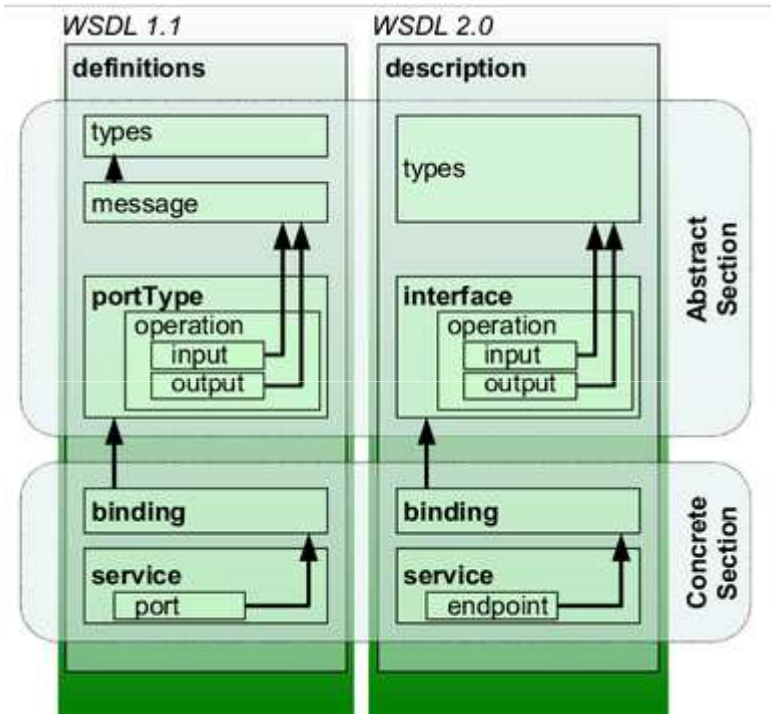
WSDL

WSDL 1.0 (Sept. 2000) has been developed by IBM, Microsoft and Ariba to describe Web Services for their SOAP toolkit.

WSDL 1.1 (March 2001) is the formalization of WSDL 1.0.

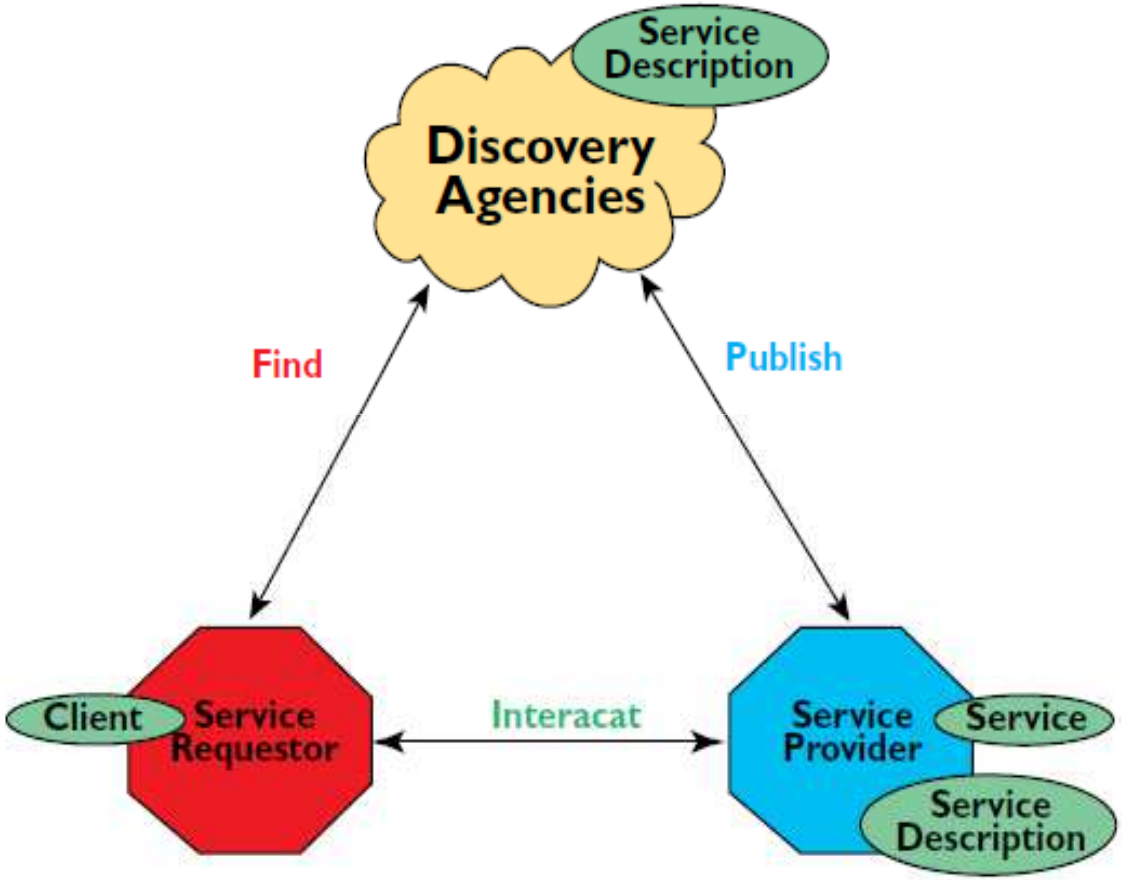
WSDL 1.2 (June 2003) attempts to remove non-interoperable features and also defined the better HTTP 1.1 binding, but is not supported by most of the SOAP servers/vendors.

WSDL 2.0 (June 2007) A simple renaming of 1.2 became a W3C recommendation



What are Web Services?

Christopher Ferris and Joel Farrell,
Commun. ACM 46, 6 (June 2003), 31



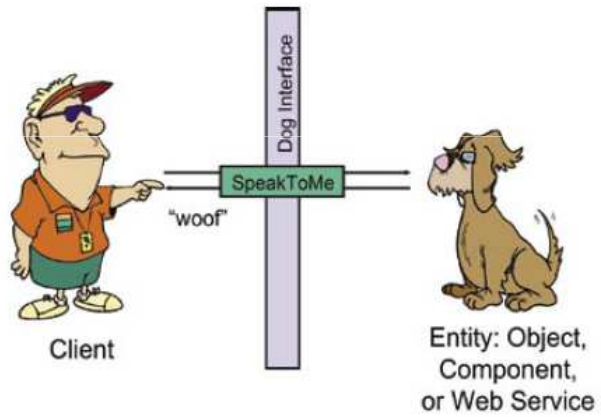
A web service, as defined by the W3C Web Services Architecture Working Group, is "*a software application identified by a URI, whose interfaces and bindings are capable of being defined, described, and discovered as XML artifacts. A Web service supports direct interactions with other software agents using XML-based messages exchanged via Internet-based protocols*"

Fuzzy Boundaries: Objects, Components, and Web Services

Roger Sessions
Queue 2, 9 (December 2004), 40-47

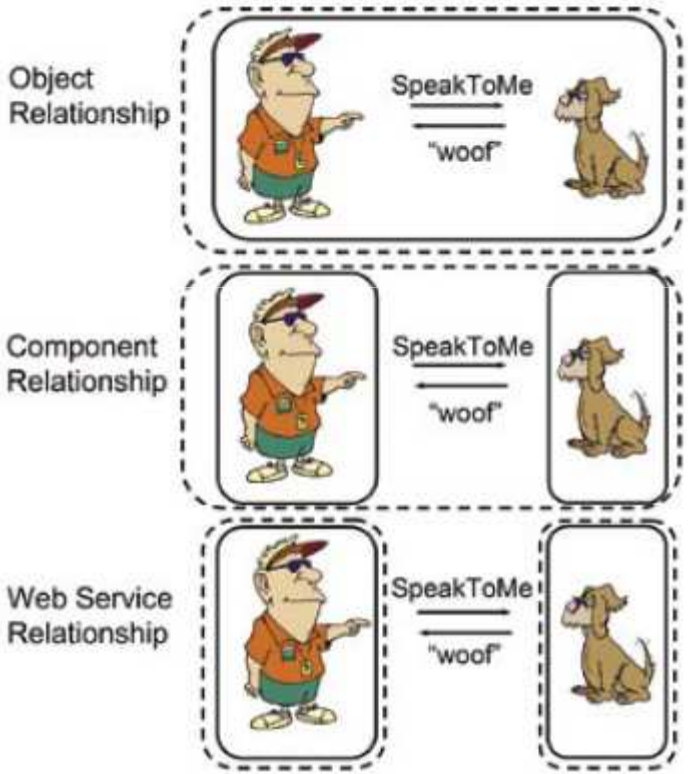
There was still much confusion so Roger Sessions tried to explain simply!

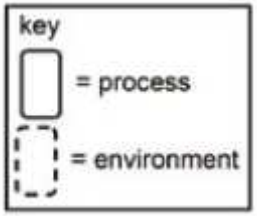
Objects, Components, and Web Services



Generic Template

Location and Environment



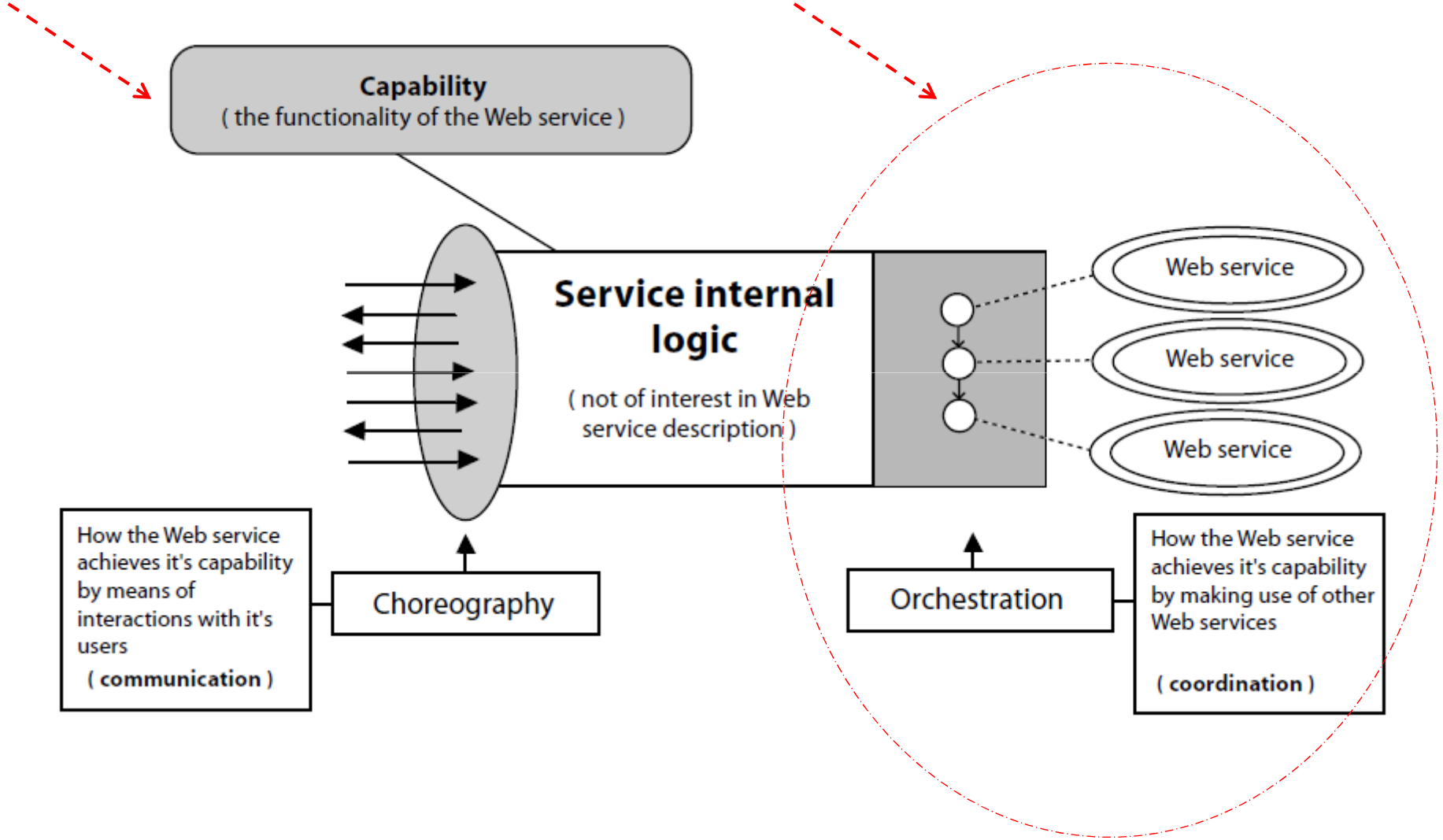


Specific Instances

Web service modeling ontology

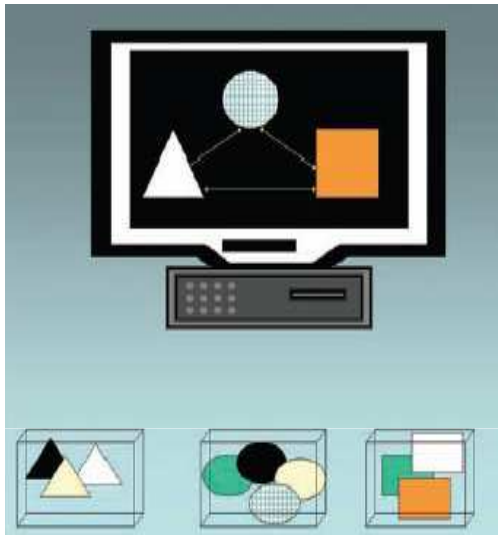
Dumitru Roman et al,
Appl. Ontol. 1, 1
(January 2005), 77-106

Move towards semantics and formal models/methods

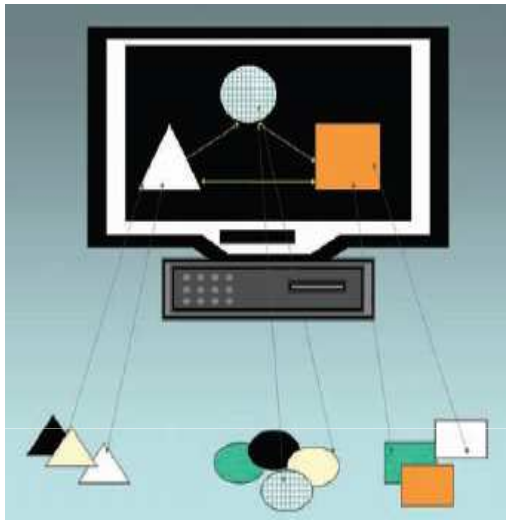


Dealing with change: components versus services

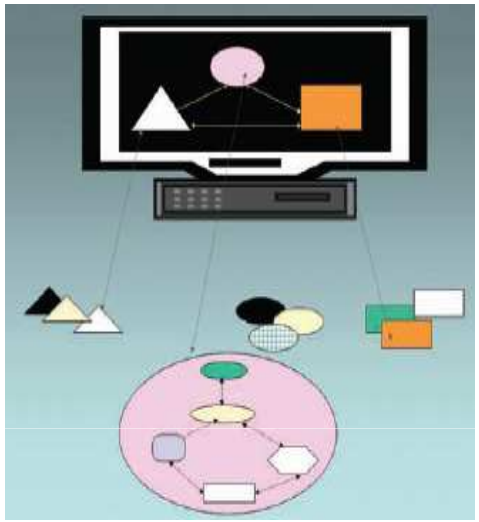
Ahmed Elfatatry, *Commun. ACM* 50, 8 (August 2007), 35-39



Component-based development.



Service-based model.



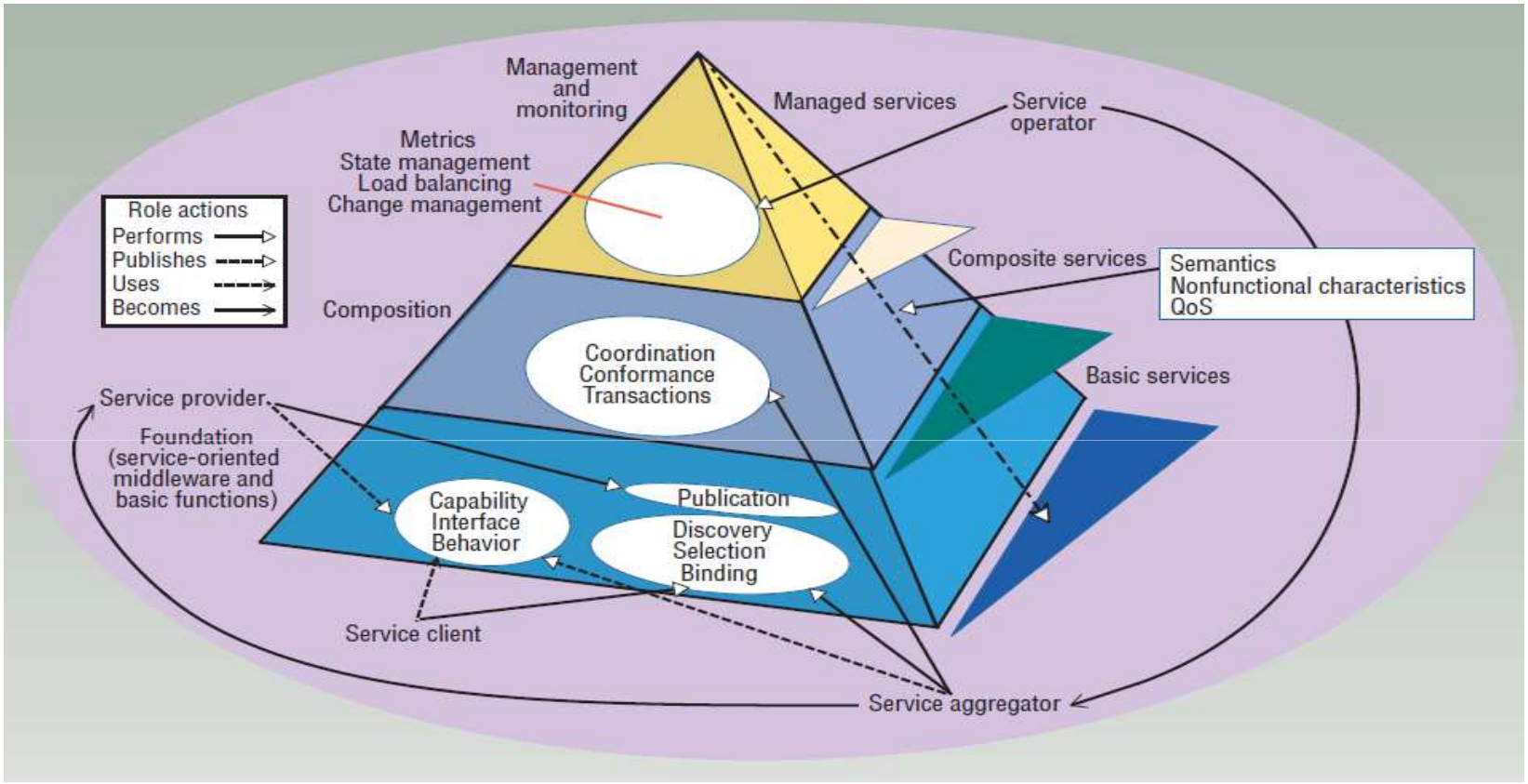
Composite service.

There was then a move towards service composition, and self* services

This is now, perhaps, the most challenging research area in service provision

Service-Oriented Computing: State of the Art and Research Challenges

Michael P. Papazoglou, Paolo Traverso, Schahram Dustdar, and Frank Leymann. *Computer* 40, 11 (November 2007), 38-45.



SOC research road map.The architectural layers provide a logical separation of functionality, while the perpendicular axis indicates service characteristics that cut across all three planes.

Note: there are lots of web service protocols

- BEEP - Blocks Extensible Exchange Protocol
- E-Business XML
- Hessian
- JSON-RPC
- Qworum
- REST (Representational State Transfer)
- SOAP
- Universal Description, Discovery, and Integration (UDDI)
- Web Processing Service (WPS)
- Web Services Description Language (WSDL)
- WSFL - Web Services Flow Language (superseded by BPEL)
- WSCL - Web Services Conversation Language
- XINS Standard Calling Convention - HTTP parameters in (GET/POST/HEAD), POX out
- XLANG - XLANG-Specification (superseded by BPEL)
- XML-RPC - XML Remote Procedure Call

HINT: you should, for now, focus on SOAP and REST

Developing web services choreography standards—the case of REST vs. SOAP

Michael zur Muehlen, Jeffrey V. Nickerson, and Keith D. Swenson, *Decis. Support Syst.* 40, 1 (July 2005), 9-29.

	REST	SOAP
Characteristics	Operations are defined in the messages Unique address for every process instance Each object supports the defined (standard) operations Loose coupling of components	Operations are defined as WSDL ports Unique address for every operation Multiple process instances share the same operation Tight coupling of components
Self-declared advantages	Late binding is possible Process instances are created explicitly Client needs no routing information beyond the initial process factory URI Client can have one generic listener interface for notifications	Debugging is possible Complex operations can be hidden behind façade Wrapping existing APIs is straightforward Increased privacy
Possible disadvantages	Large number of objects Managing the URI namespace can become cumbersome	Client needs to know operations and their semantics beforehand Client needs dedicated ports for different types of notification Process instances are created implicitly

The simplest (?) web service: read a web page contents from its url

http://www.vogella.de/articles/JavaNetworking/article.html

```
package de.vogella.web.html;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URL;

public class ReadWebPage {
        public static void main(String[] args) throws IOException {
                String urltext = "http://www.vogella.de";
                URL url = new URL(urltext);
                BufferedReader in = new BufferedReader(new InputStreamReader(url
                                .openStream()));
                String inputLine;

                while ((inputLine = in.readLine()) != null) {
                        // Process each line.
                        System.out.println(inputLine);
                }
                in.close();
        }
}
```

QUESTION: (how) does this work?

What about get services?

http://www.vogella.de/articles/JavaNetworking/article.html

Several websites offer services via Http get calls. For example your can send a get request to "http://tinyurl" or http://tr.im" and receive a short version of the Url you pass as parameter.

```
package de.vogella.web.get;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URL;

public class TinyURL  {
        private static final String tinyUrl = "http://tinyurl.com/api-create.php?url=";

        public String shorter(String url) throws IOException {
                String tinyUrlLookup = tinyUrl + url;
                BufferedReader reader = new BufferedReader(new InputStreamReader(new URL(tinyUrlLookup).openStream()));
                String tinyUrl = reader.readLine();
                return tinyUrl;
        }


}
```

QUESTION: (how) does this work?

Problem Based Learning: Composing Web Services

Using Java, you are to build a simple program that illustrates how to use the web to provide re-usable functionality:

•The architecture is a simple pipeline

•The user inputs a string.

•The Java program outputs a string

•There must be at least 2 « web components/services » used in the internal processing: