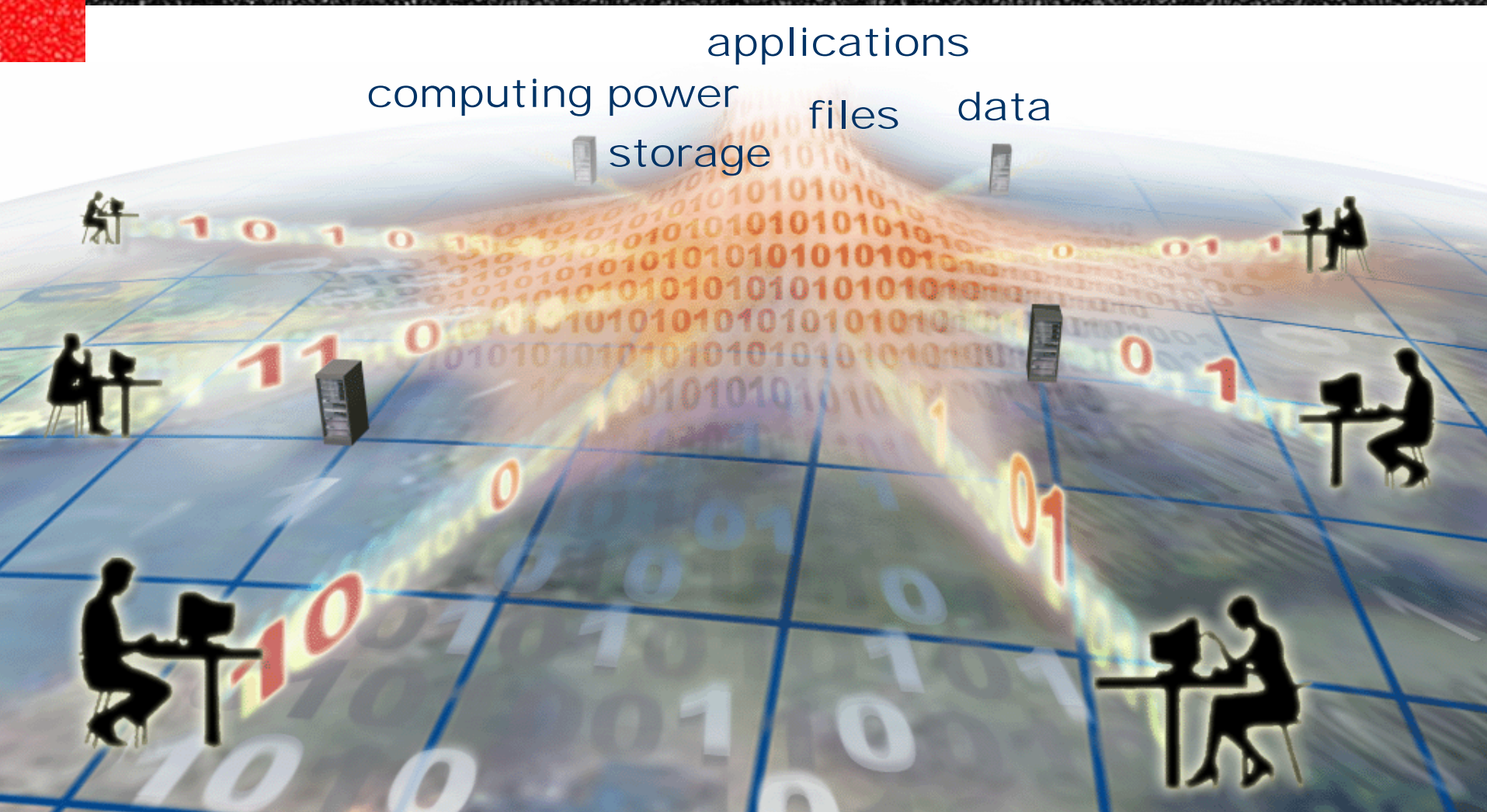


# Service Proxying with Dependency Injection

SORCER Seminar  
Sept 28, 2006

**Michael Sobolewski**  
Computer Science, TTU  
sobol@cs.ttu.edu  
<http://sobol.cs.ttu.edu>

# From the Internet ....



applications

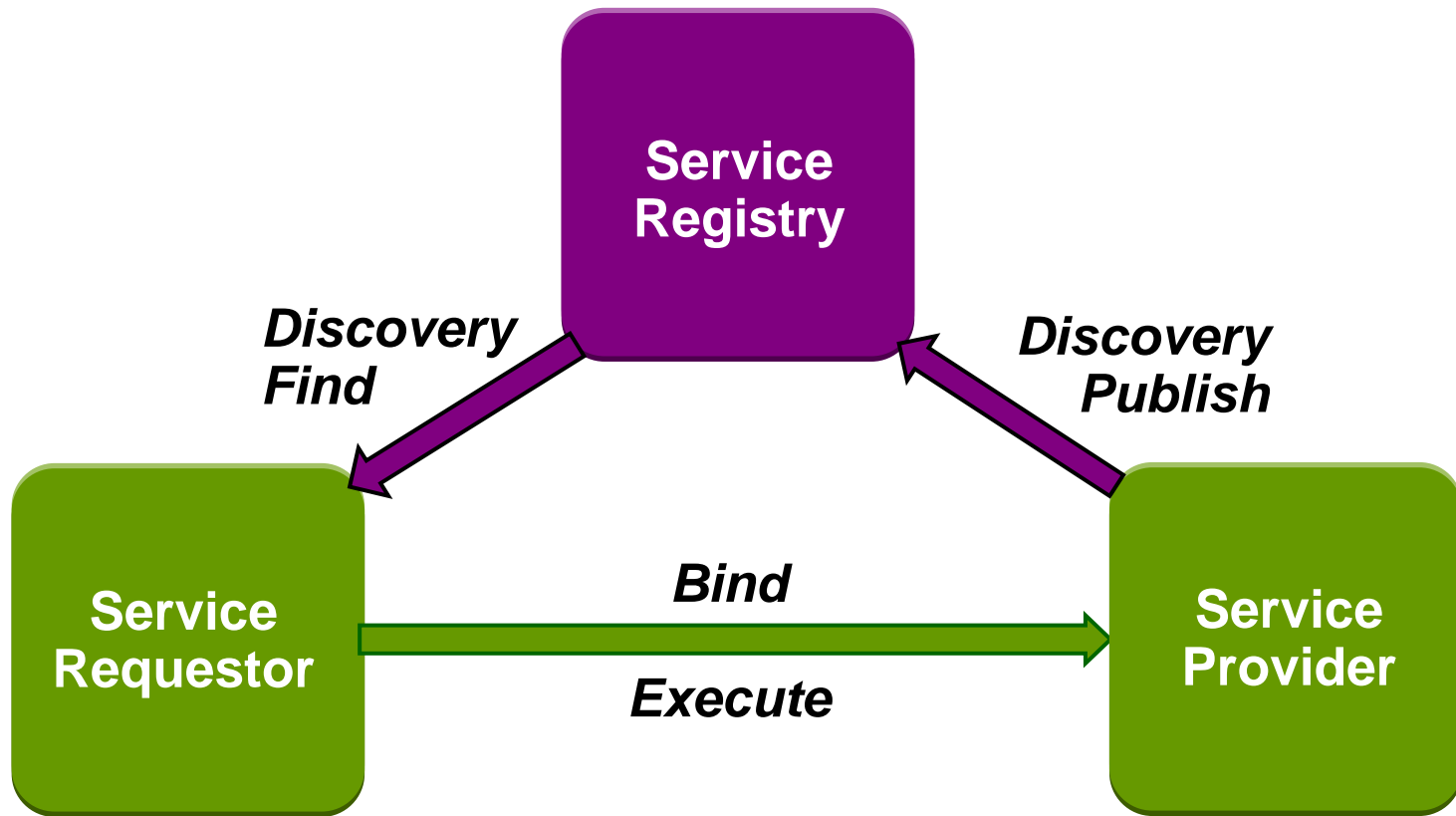
computing power files data  
storage



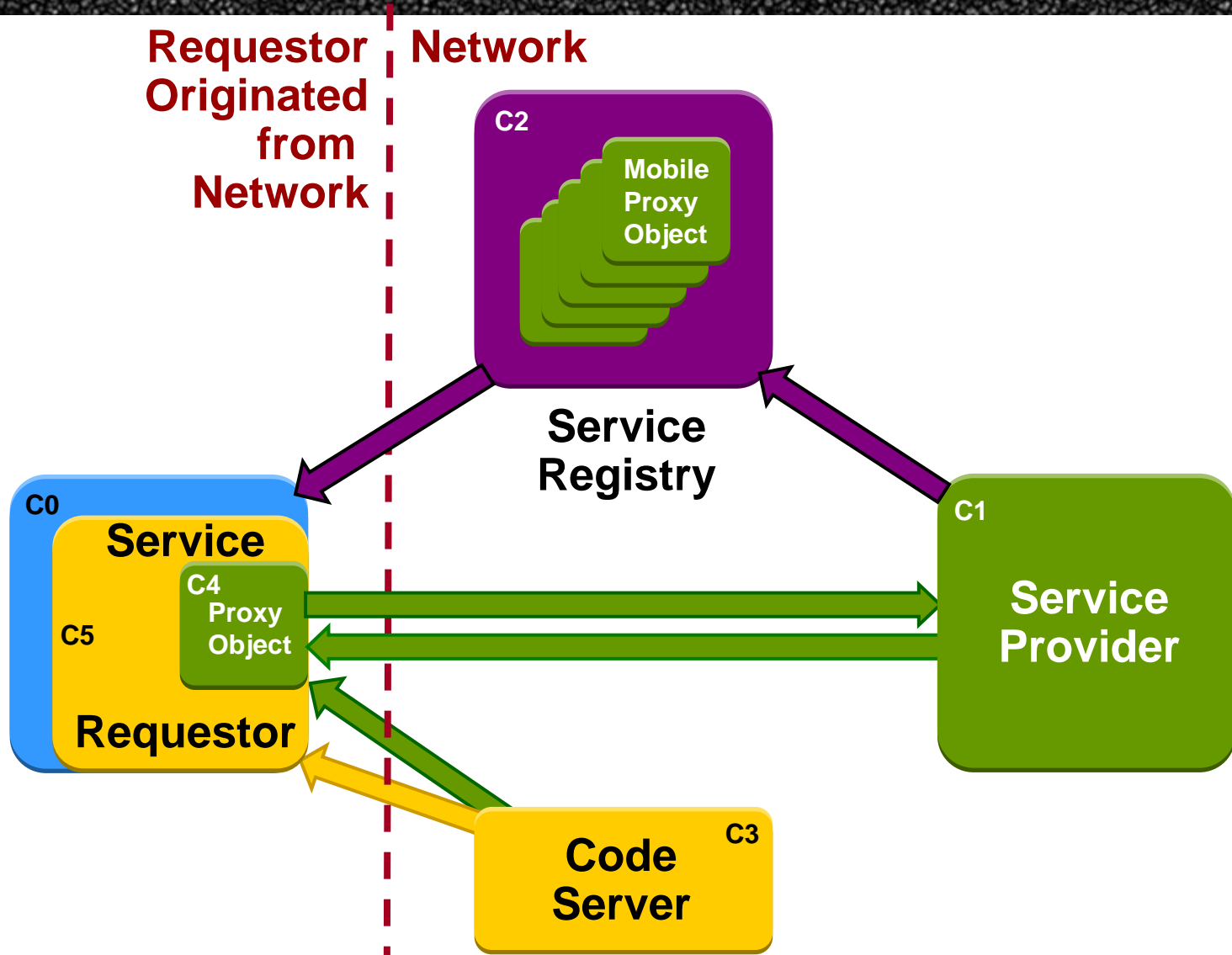
# .... to Metacomputing



# SOA = SPOA + SOOA



# SOOA - Network Centricity

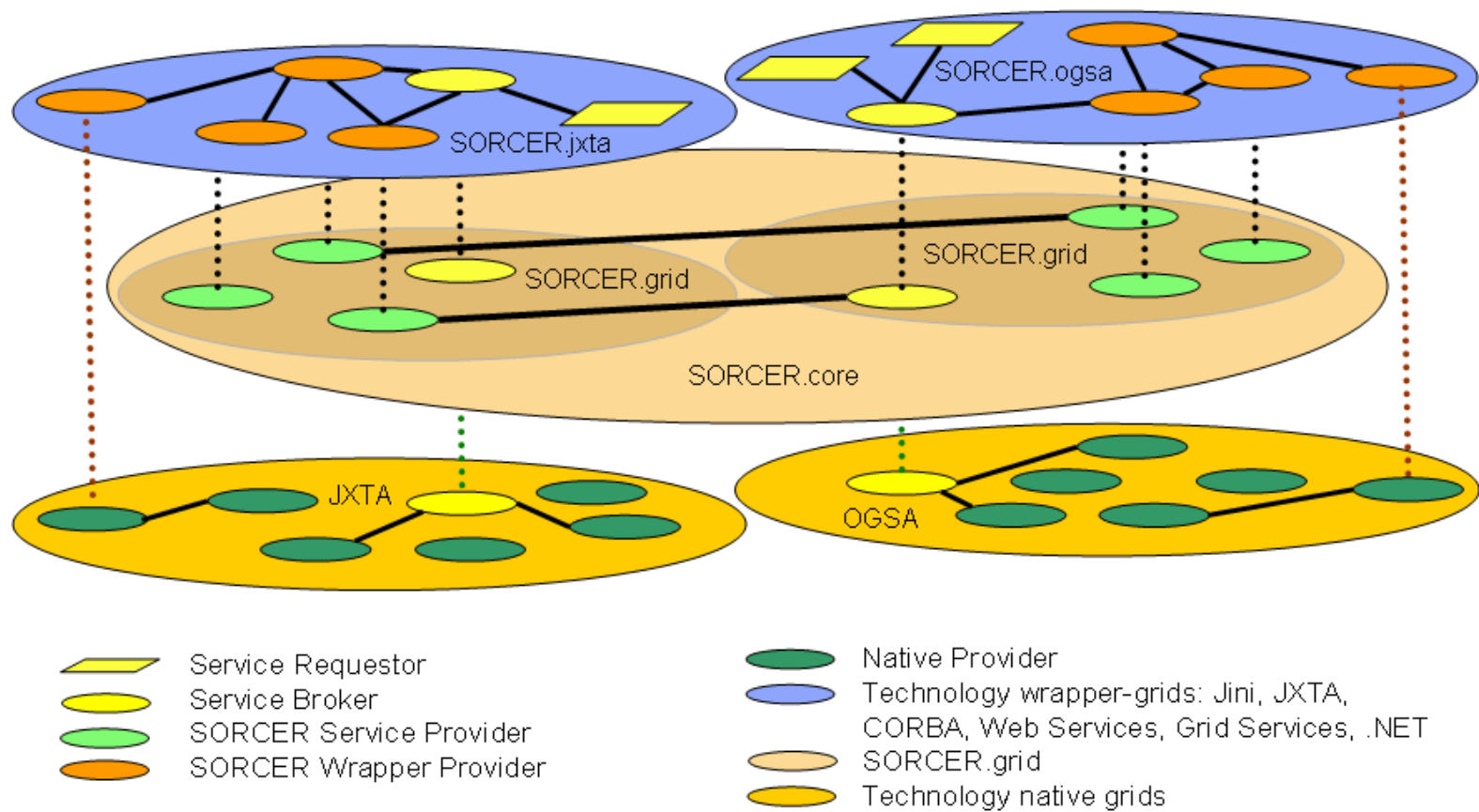


# Proxy Types

1. Static proxy – created explicitly before it is used (rmic)
2. Dynamic proxy – no need to create it statically in advance (RMI vs. Jini ERI)
3. Remote proxy – shields the requestor object from the fact that the underlying object is remote
4. Access (protection) proxy – enforces a security policy on access to a service or data-providing object (Sevicer)
5. Façades – a façade grants access to multiple underlying objects (Sevicer + AdminProxy + server)
6. Virtual proxy – performs lazy initialization of expensive back-ends (Service UI – UIDescriptor)
7. Smart proxy - grants access to local (fat) and remote resources
8. Bootstrap proxy – trusted proxy, created from local codebase (proxy verification)
9. Hybrid proxy – combination of the above types



# Intergrid



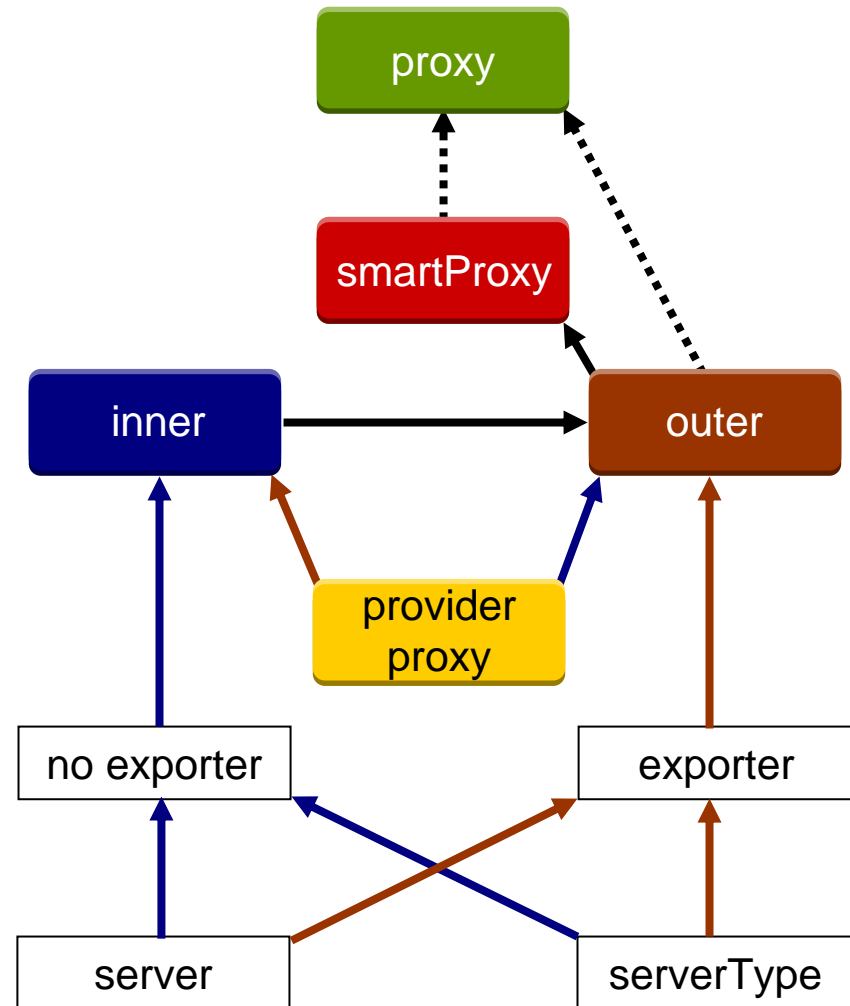
# The Runtime Environment

- SORCER (Eclipse) workspace
- Jini services (usually available on the network)
  - Lookup service (reggie)
  - JavaSpace (outrigger)
  - Transaction Manager (mahalo)
  - Event Mailbox (mercury)
  - Lease Renewal (norm)
  - Lookup Discovery (fiddler)
- Webster (HTTP class server) - iGrid/bin/webster/bin
- Service browser (Inca X) - iGrid/bin/incax/bin
- SORCER services - iGrid/bin/sorcer/bin
- Custom services - iGrid/bin/<serviceName>/bin



# Proxying with Dependency Injection

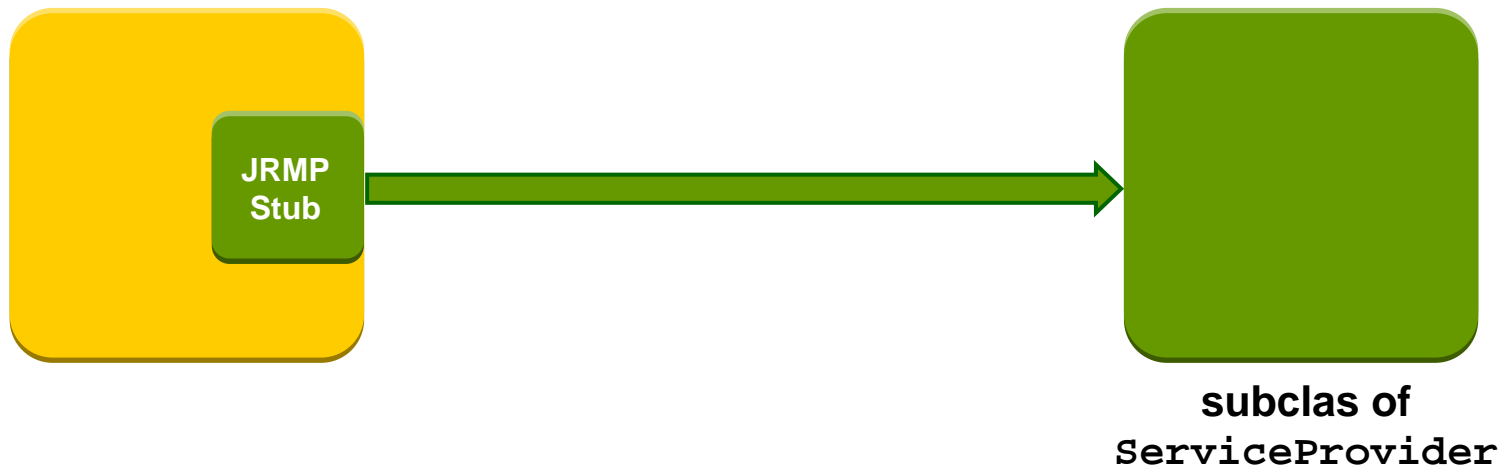
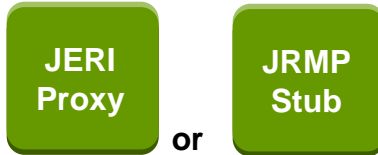
- smartProxy
- server
- serverType
- serverExporter
- beans



Twelve cases studied



# Providers Implementing Remote Interfaces



**Provider = Server**  
Direct calls can be forbidden with indirect service calls  
`Service#service(Exertion):Exertion`



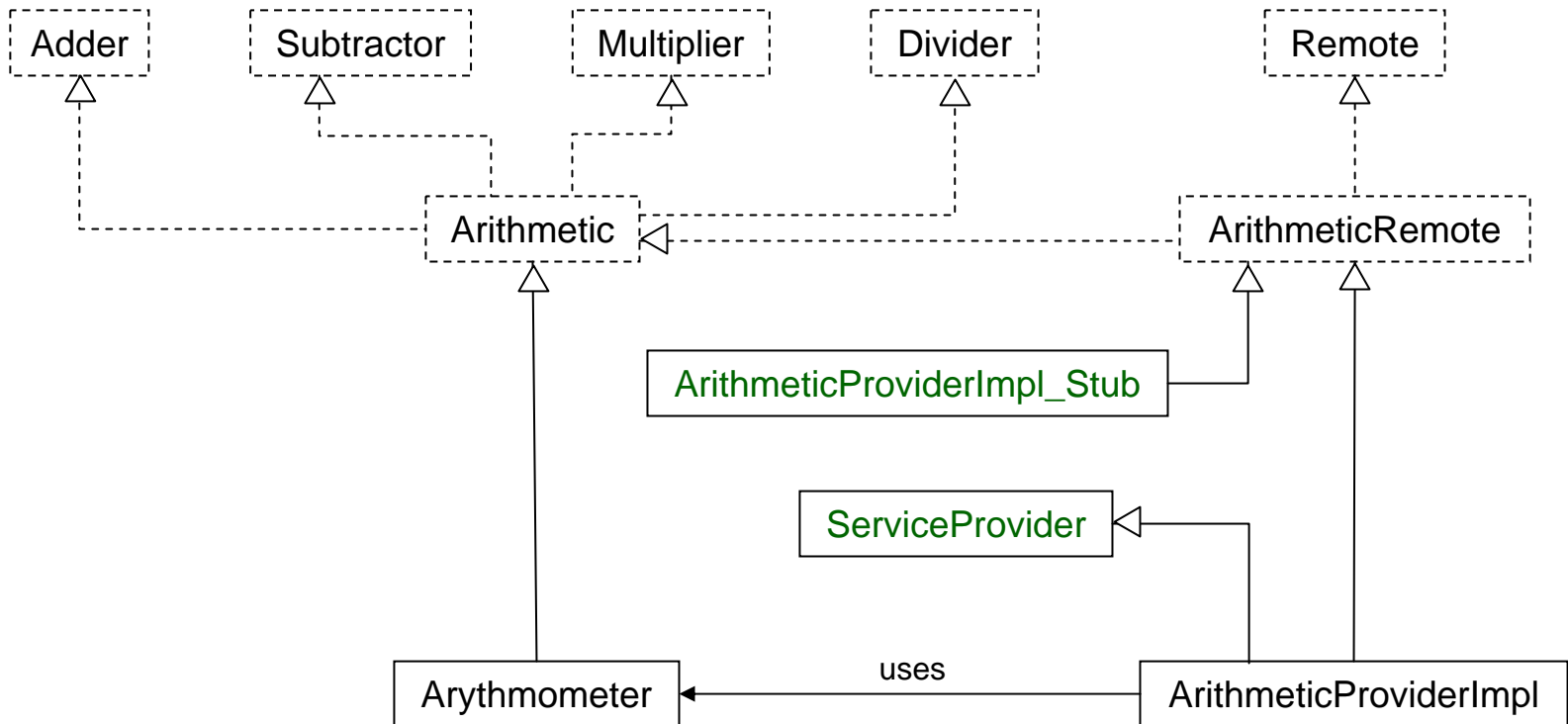
# ArithmeticImpl Implements ArithmeticRemote

JERI  
Proxy

or

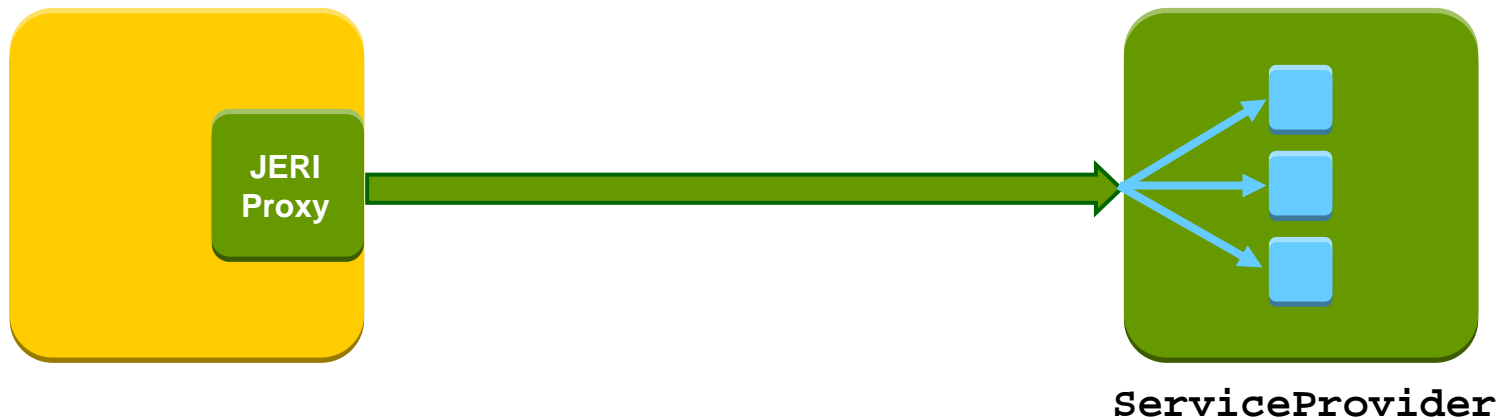
JRMP  
Stub

*jeri and jrmp*



# Providers with Service Beans

JERI  
Proxy



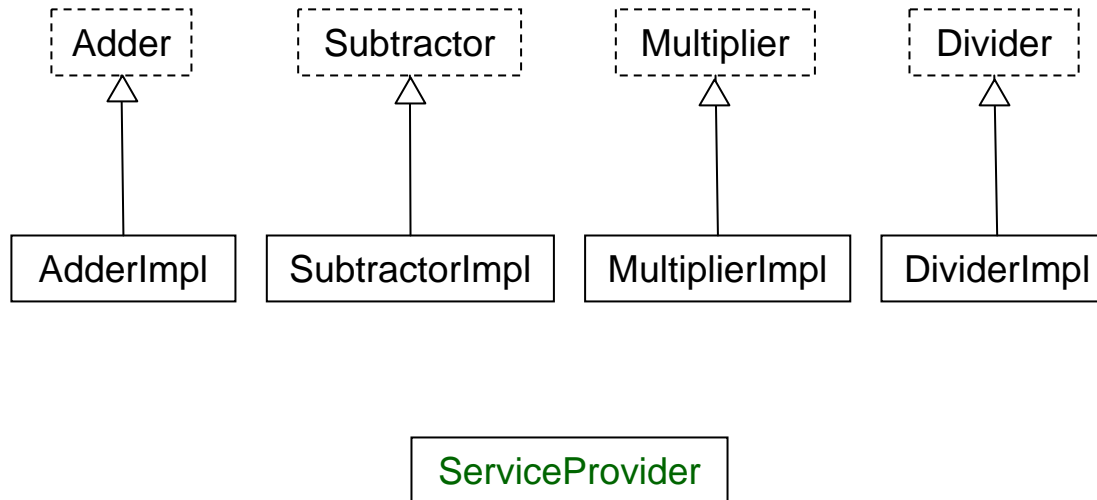
**Provider = ServiceProvider + beans;  
bean - implements service methods in its exposed (not Remote)  
interfaces.**

**Beans are not servers, they are not exported.**

# Service Beans

JERI  
Proxy

*beans*

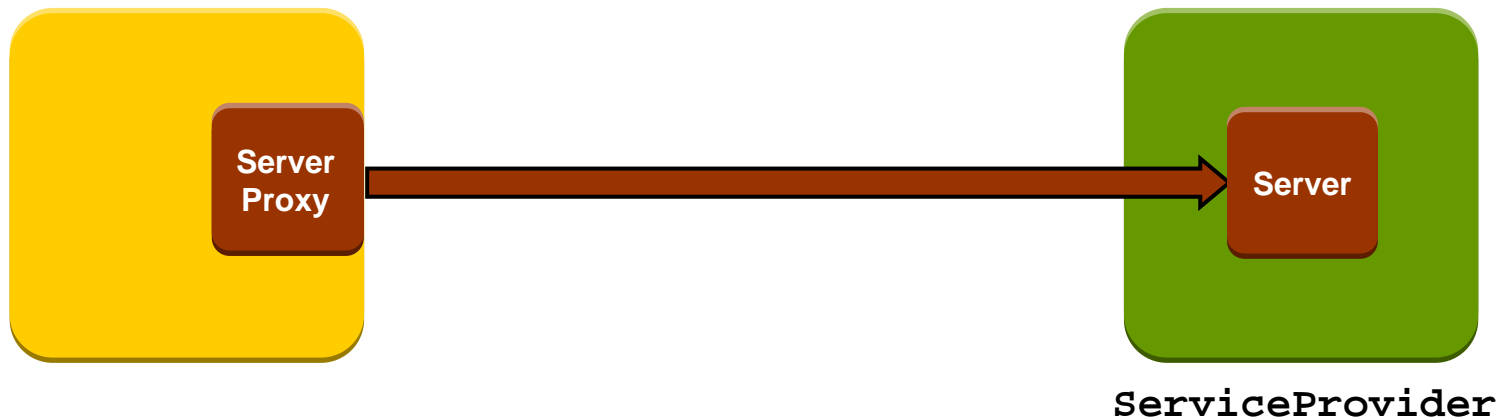


```
beans = new String[] {
    "sorcer.arithmetic.AdderImpl",
    "sorcer.arithmetic.SubtractorImpl",
    "sorcer.arithmetic.MultiplierImpl",
    "sorcer.arithmetic.DividerImpl" };
```



# Providers with Exported Servers

Server  
Proxy

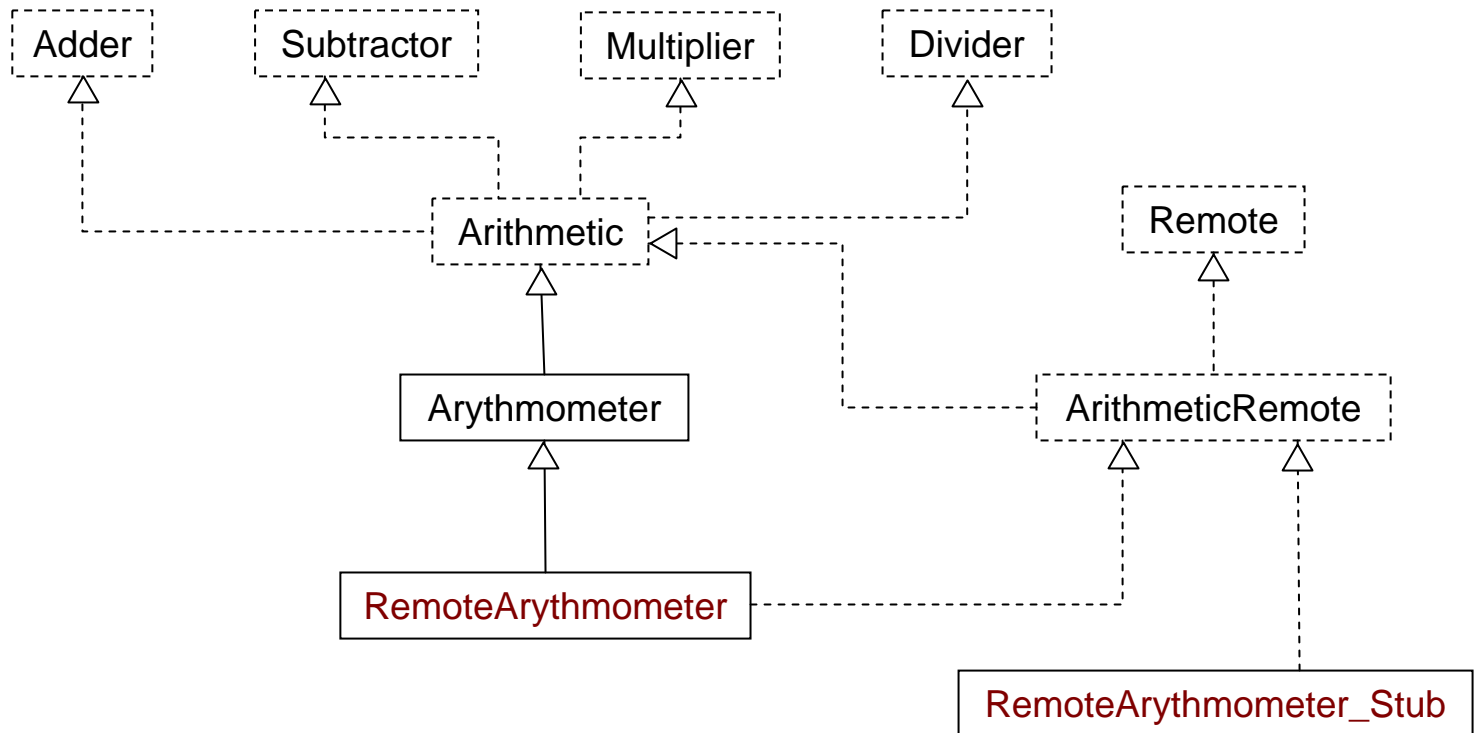


**Provider = ServiceProvider + server;  
server and serverExporter entries defined, and  
server is not Partner type**

# RemoteArithmometer Implements ArithmeticRemote

Server  
Proxy

server



```
// RMI object
server = new RemoteArithmometer();
// exported with
serverExporter = new JrmpExporter(0);
```

ServiceProvider



# Providers with Exported Partnership Servers

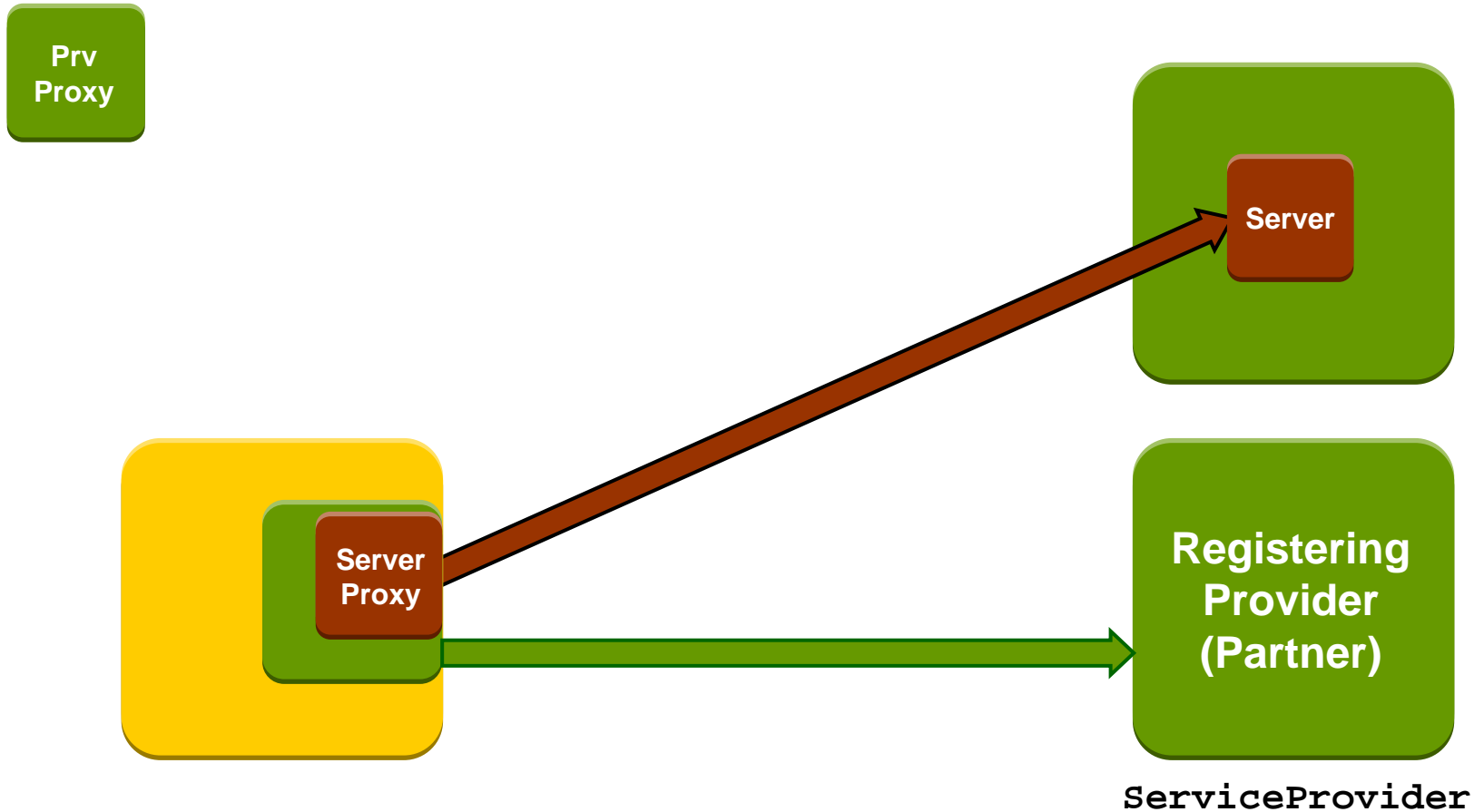
Server Proxy



**Provider = ServiceProvider + server;  
server and serverExporter entries defined, and  
server implements Partner**



# Providers with not Exported Servers



**Provider = ServiceProvider + server;**  
**server entry defined; no serverExporter entries defined, and**  
**server is not Partner type**



# Smart Proxies – Fat Proxy

Smart  
Proxy



Registering  
Provider

ServiceProvider

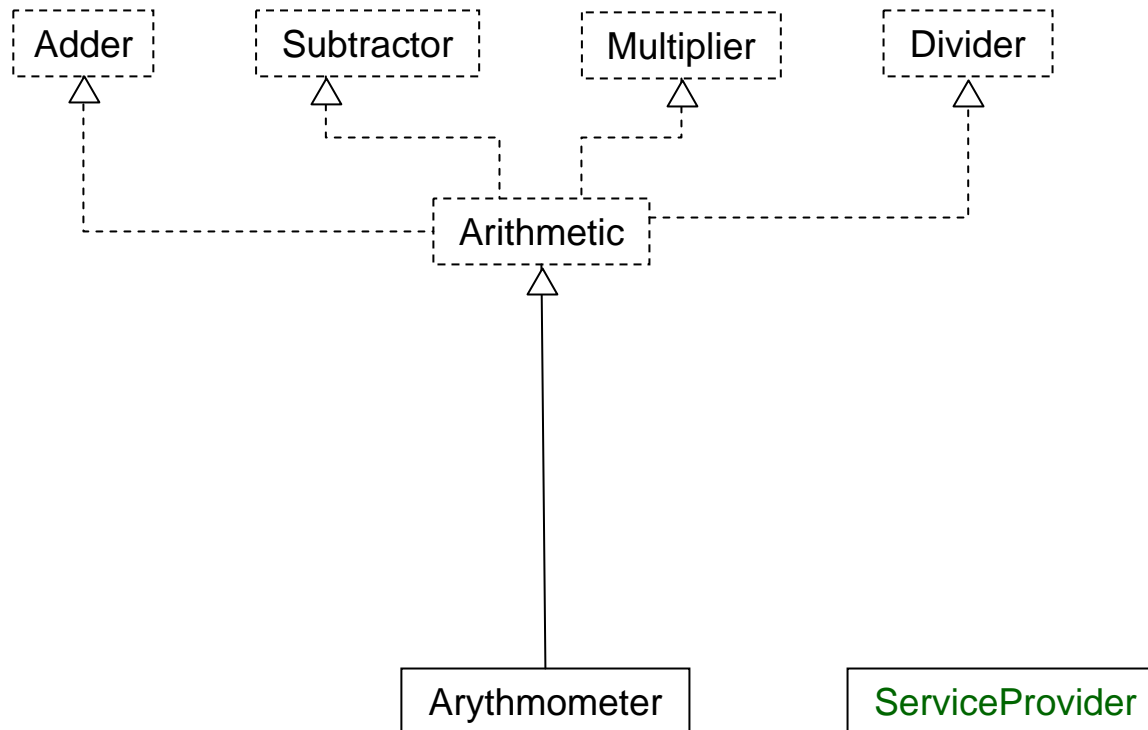
**Provider = ServiceProvider + smartProxy;  
requestor invokes local calls only;  
smartProxy is NOT Partnership type, and  
ServiceProvider maintains the proxy registration.**



# Arithmometer Implements Arithmetic (no Remote)

Smart Proxy

fat



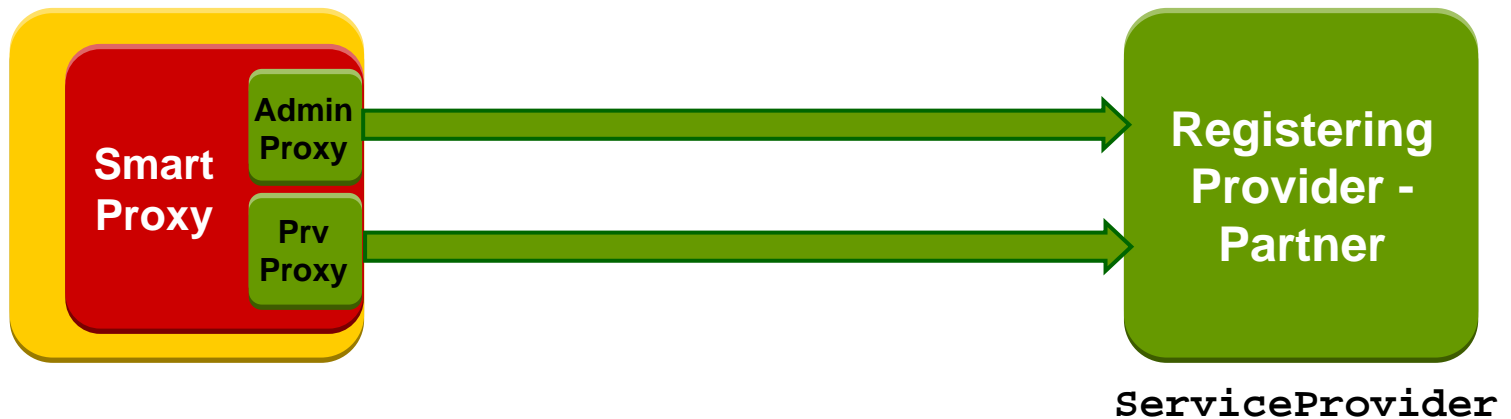
```
smartProxy = new Arithmometer();
```



# Outer Smart Proxies

Smart  
Proxy

## *Semismart Proxies*



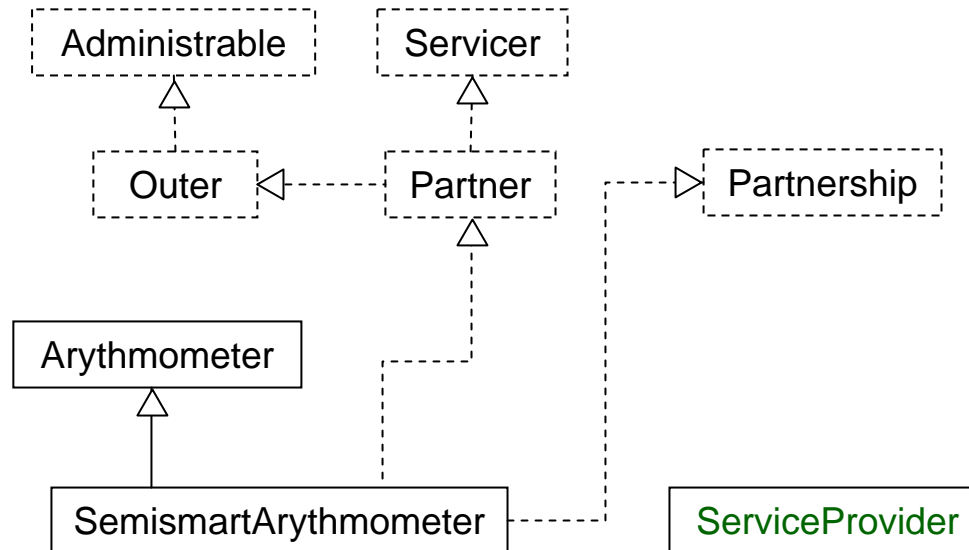
`Provider = ServiceProvider + smartProxy`  
Requestor invokes local calls only;  
`smartProxy` implements Partnership, and  
`ServiceProvider` maintains the proxy registration.



# SemismartArithmometer Implements Outer

Smart  
Proxy

*semismart*

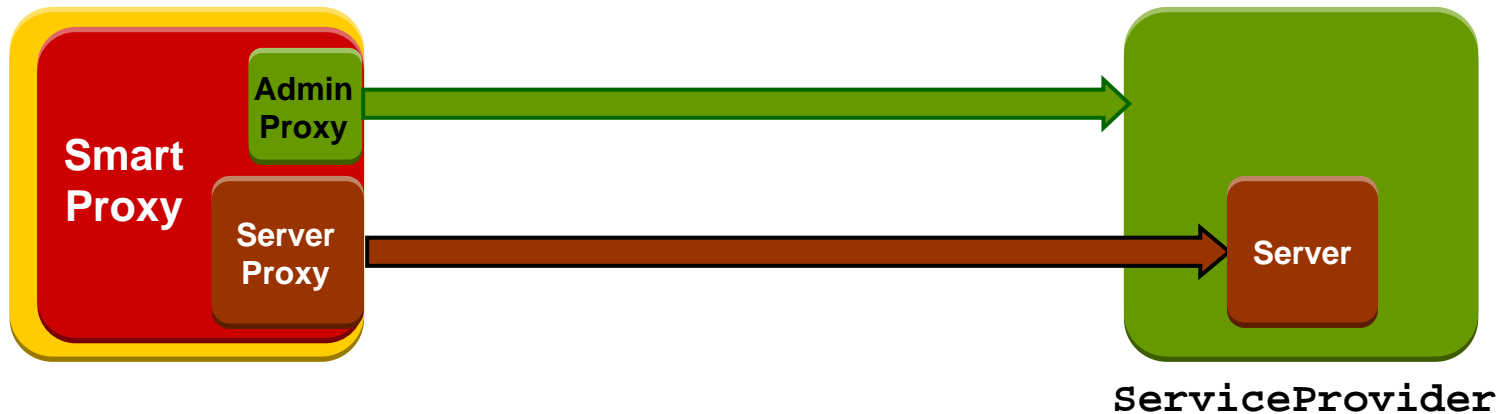


```
smartProxy = new SemismartArithmometer();
```



# Smart Proxies with Exported Servers

Smart  
Proxy

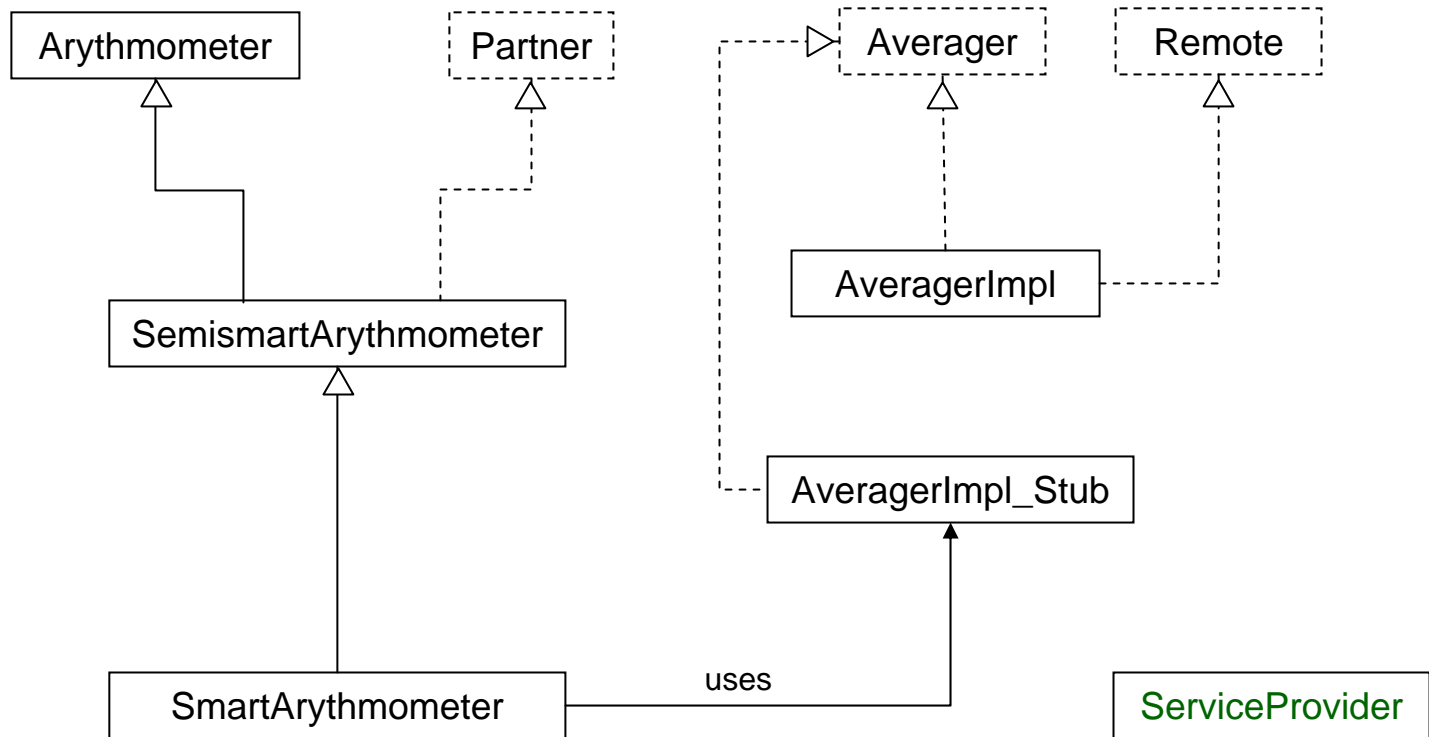


**Provider = ServiceProvider + smartProxy + server;  
server and serverExporter entries defined, and  
server is not Partnership type;  
smartProxy implements Partnership, and  
ServiceProvider maintains the smartProxy registration.**

# SmartArithmometer Implements Averager

Smart Proxy

smart



```
smartProxy = new SmartArithmometer();
server = new AveragerImpl();
serverExporter = new JrmpExporter(0);
```



# Smart Proxies with Partnership Servers

Smart Proxy

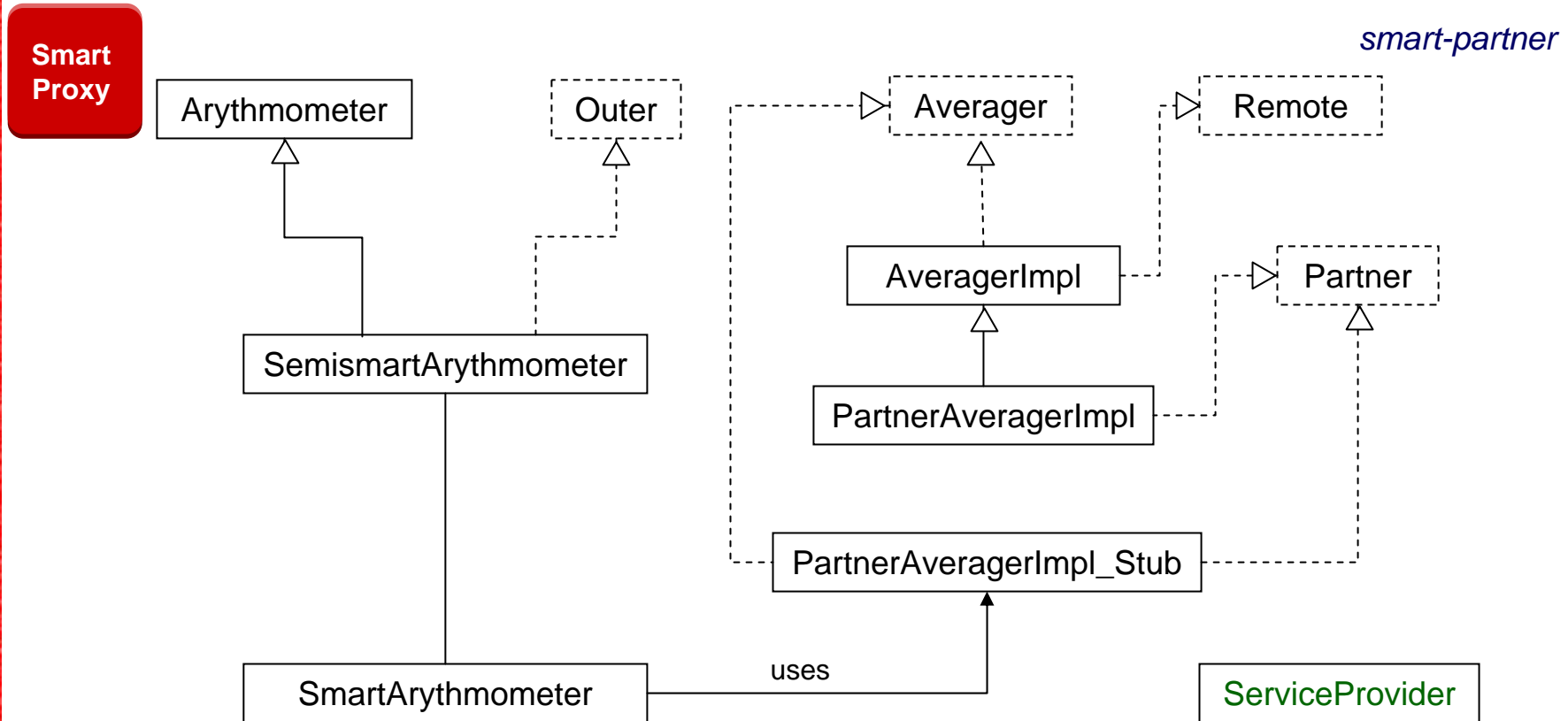


**Provider = ServiceProvider + smartProxy + server ;**  
**requestor invokes local calls only;**  
**server and serverExporter entries defined, and**  
**server implements Partnership**  
**smartProxy implements Outer, and**

**ServiceProvider maintains the smartProxy registration.**



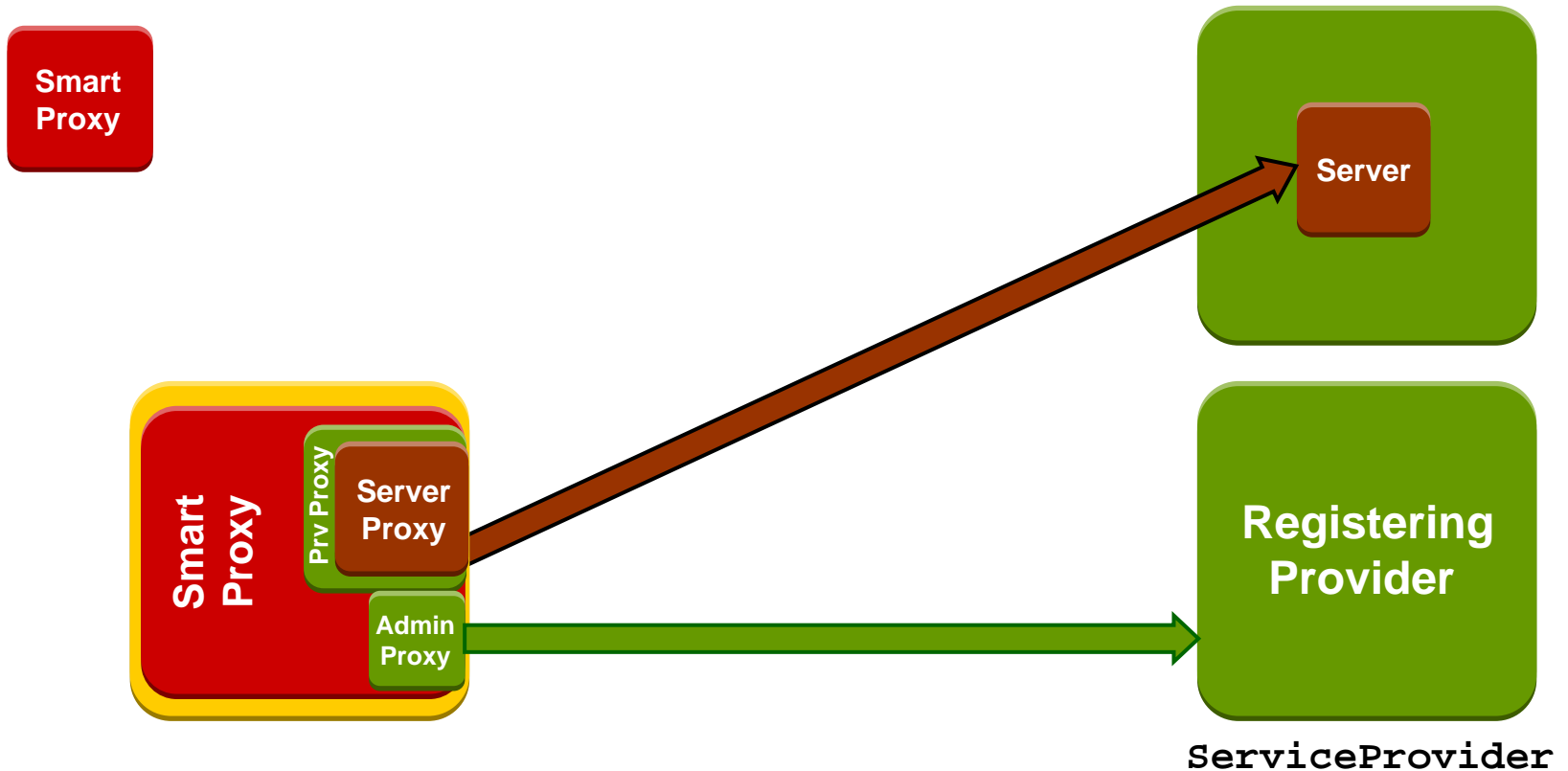
# SmartArithmometer Implements Averager, Averager Implements Partnership



```
smartProxy = new SmartArithmometer();
server = new PartnerAveragerImpl();
serverExporter = new JrmpExporter(0);
```



# Smart Proxies with not Exported Servers



`Provider = ServiceProvider + smartProxy + server;`  
`server` entry defined; no `serverExporter` entries defined, and  
`server` is Partnership type;  
`smartProxy` implements `Outer`, and  
`ServiceProvider` maintains the `smartProxy` registration.



# Proxying with Taskers

*tasker*

Tasker  
Proxy



*Tasker Task* (`ArithmeticTask`)  
*Tasker Method* (`ArithmeticMethod`)

`ServiceProvider`  
 implements `Tasker`

**Tasker executes inserted *Tasker Method***



# Proxying with Callers

*caller*

Caller  
Proxy



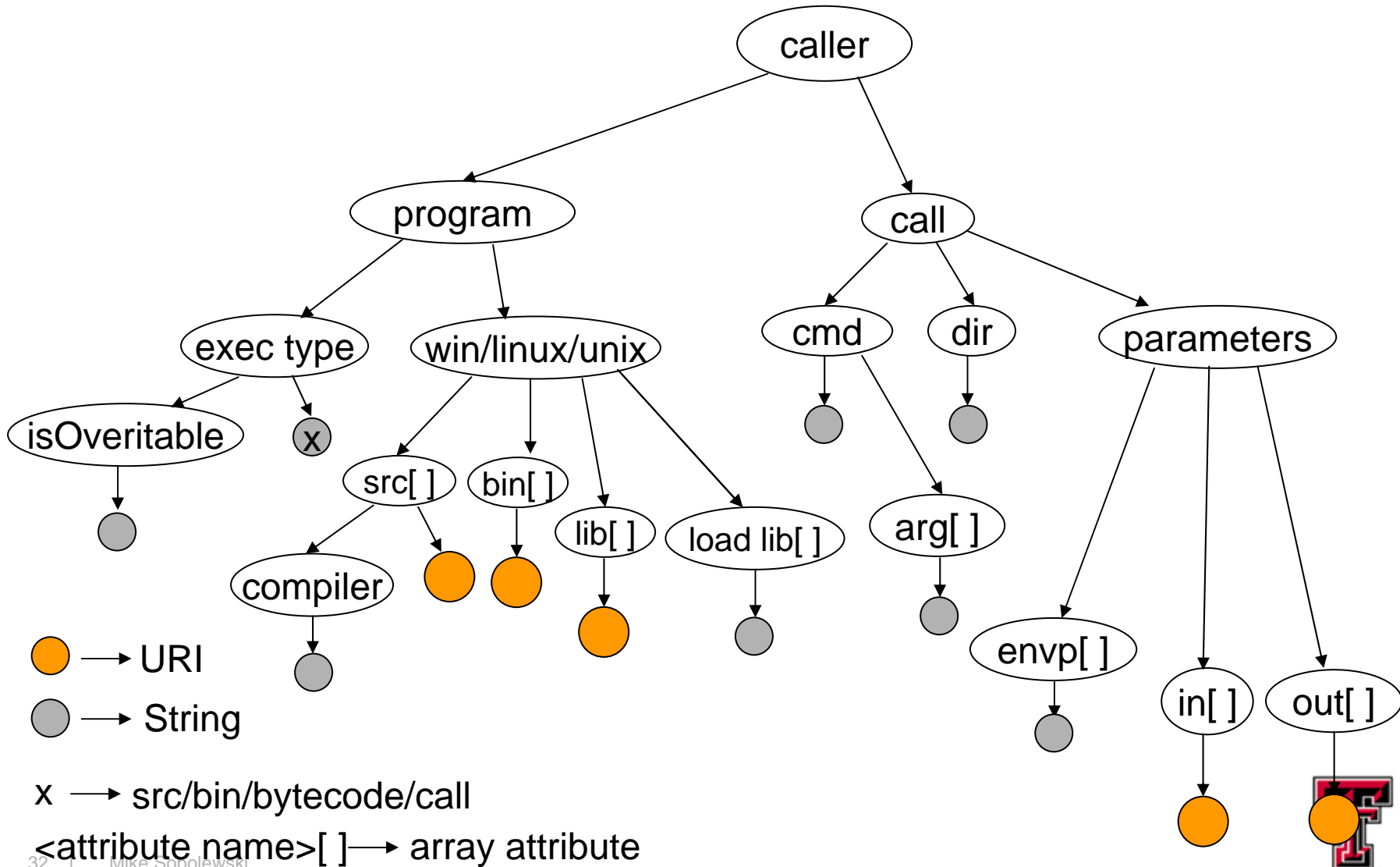
***Caller Task with CallerContext***

**ServiceProvider  
implements Caller**

**Callers make a context-based system call**



# Caller Service Context



# SORCER Research Domain

- **Service-Oriented Programming**
- **Service-Oriented Computing Environment**
- **Service-Oriented Programming Development Tools**
- **Service-Federated Assurance and Security**
- **Self-Aware Service Federations**
- **Autonomic Service Federations**
- **Service Federated Intergrids**
- ***Metacomputing Service-Oriented MAO***
- **SORCER Theses**



# Q&A