Provisioning Object-oriented Service Clouds for Exertion-oriented Programming

Michael Sobolewski

SORCERsoft.org, AFRL/WPAFB

May 9, 2011

Agenda

- Intro: computing science & process expression
- Distribution, object & service orientation
- Transdisciplinary computing processes — SO Platform
- C/S, SOA, SPOA, SOOA and **FSOOA**
- SORCER metaprogramming and programming
  - **EOL**, **VOL**, **VML**
- SORCER Operating System (**SOS**) and **FMI**
- SORCER Virtual Processor and **Provisioning**
- Conclusions

Whatever we may want to say,
*we probably **won't say exactly that**!*

Marvin Minsky

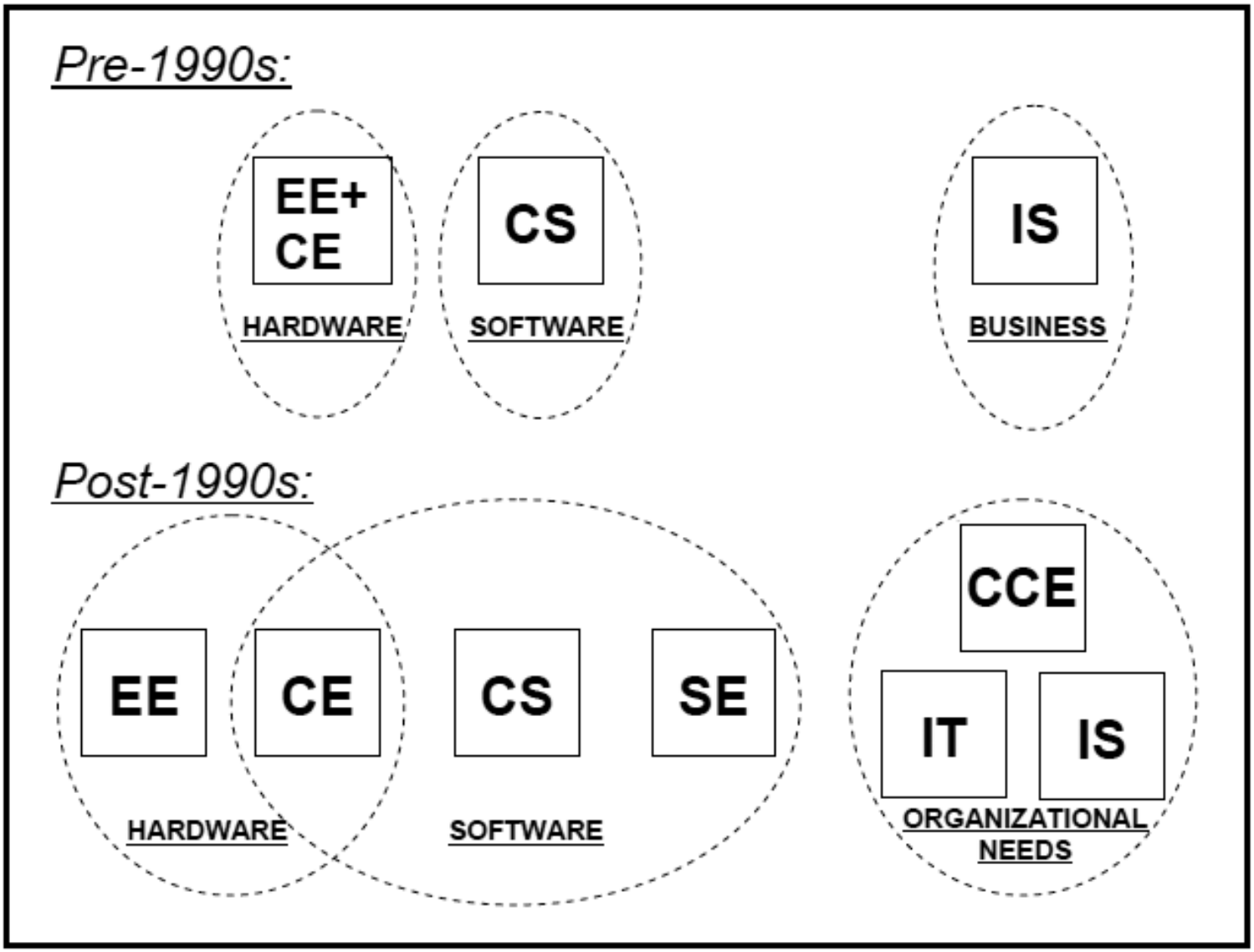
From AI to Metacomputing

- AI/Expertalk (PAN/PW/UŚ) – 1971-1989
- DICEtalk (CERC/WVU, DICE/DARPA) – 1989-1994
- CAMnet (GE GRC/DARPA) – 1994-1995
- Agile Castings (GE GRC/DARPA) – 1995-1998
- WCE: UNS Notebook (GE GRC/AE), GE Plastics Calculator, EMPIS (GE PS), ++, – 1997-2000
- FIPER (GE GRC/NIST) – 1999-2003
- SORCER
  (TTU – 2002-2009, SORCERsoft/AFRL/++) – 2007-…

Computing Science

Computing's core challenge is how ***not to make a mess*** *of it.*

Edsger Dijkstra



Process Expression

- *Computer science is the science of process expression*

  Karl Fant

- Process expression
  - Symbolic expression (Language)
    - grammar or metamodel (UML Behavior Diagrams)
  - Physical expression (Actualization)
    - computing platform
      - Mogramming environment
      - Operating system (Command, OO, SO)
      - Processor (native or virtual)

Process Expression

- Persian abacus (600 BC)
- Algorism (alKhowarizmi, 825)
  (algorists vs. abacists)
- Mathematics (Hilbert, 1920)
  (expressions complete, consistent, decidable)
- Algorithm – flowchart (Markov, 1954)
- Case-based, rule-based, and connectionist expressions
- Logic circuits (programmable FPGA, FPAA)
- Object interaction (object orientation)
- MOF/UML (M2, behavior diagrams)
- Service federation (service-orientation—exertions)

*You don't understand anything until you **learn it more than one way**.*

Marvin Minsky

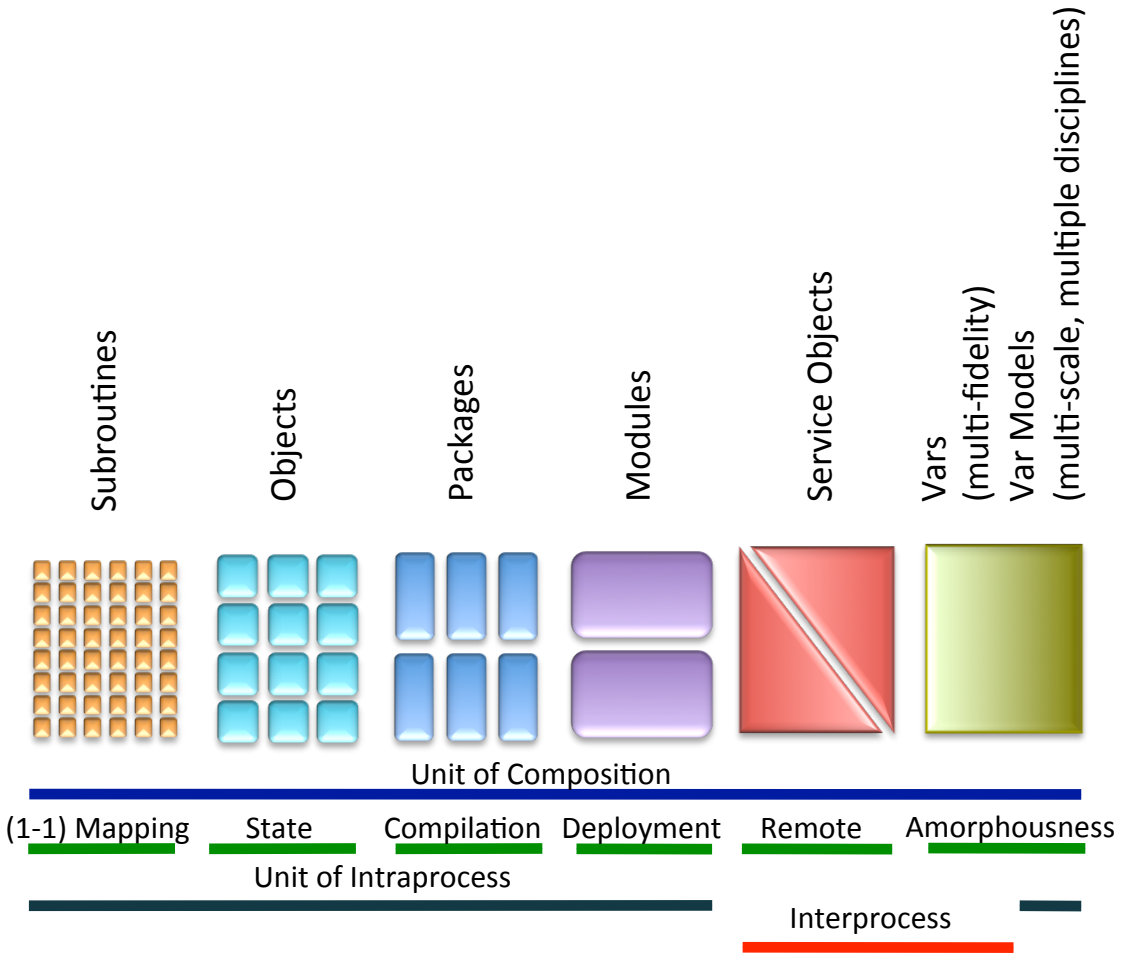
Agenda

- Intro: computing science & process expression
- Distribution, object & service orientation
- Transdisciplinary computing processes — SO Platform
- C/S, SOA, SPOA, SOOA and **FSOOA**
- SORCER metaprogramming and programming
  - **EOL**, **VOL**, **VML**
- SORCER Operating System (**SOS**) and **FMI**
- SORCER Virtual Processor and **Provisioning**
- Conclusions

Terminology & Technology

- "The computer is the network." vs. "The network is the computer." (eight fallacies of network computing)
  - MS/IBM: The network is the App server.
  - Oracle: The network is the database.
- "A distributed OO system" vs. "An OO distributed system"
  - A distributed OO system – implicit network (network transparency)
  - An OO distributed system – explicit network

Composition Granularity



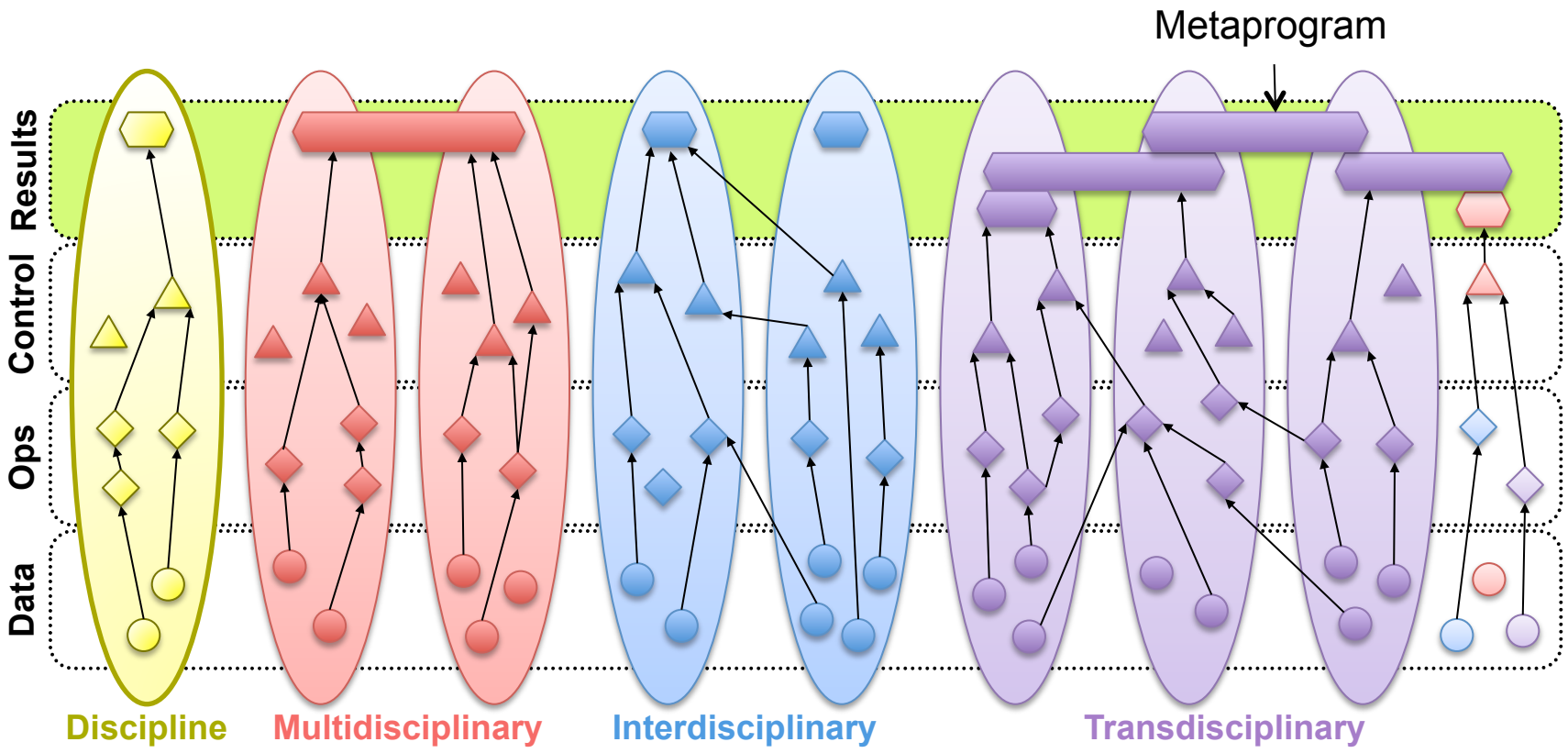
Abstractions of Programming Components

Agenda

- Intro: computing science & process expression
- Distribution, object & service orientation
- Transdisciplinary computing processes — SO Platform
- C/S, SOA, SPOA, SOOA and **FSOOA**
- SORCER metaprogramming and programming
  - **EOL**, **VOL**, **VML**
- SORCER Operating System (**SOS**) and **FMI**
- SORCER Virtual Processor and **Provisioning**
- Conclusions

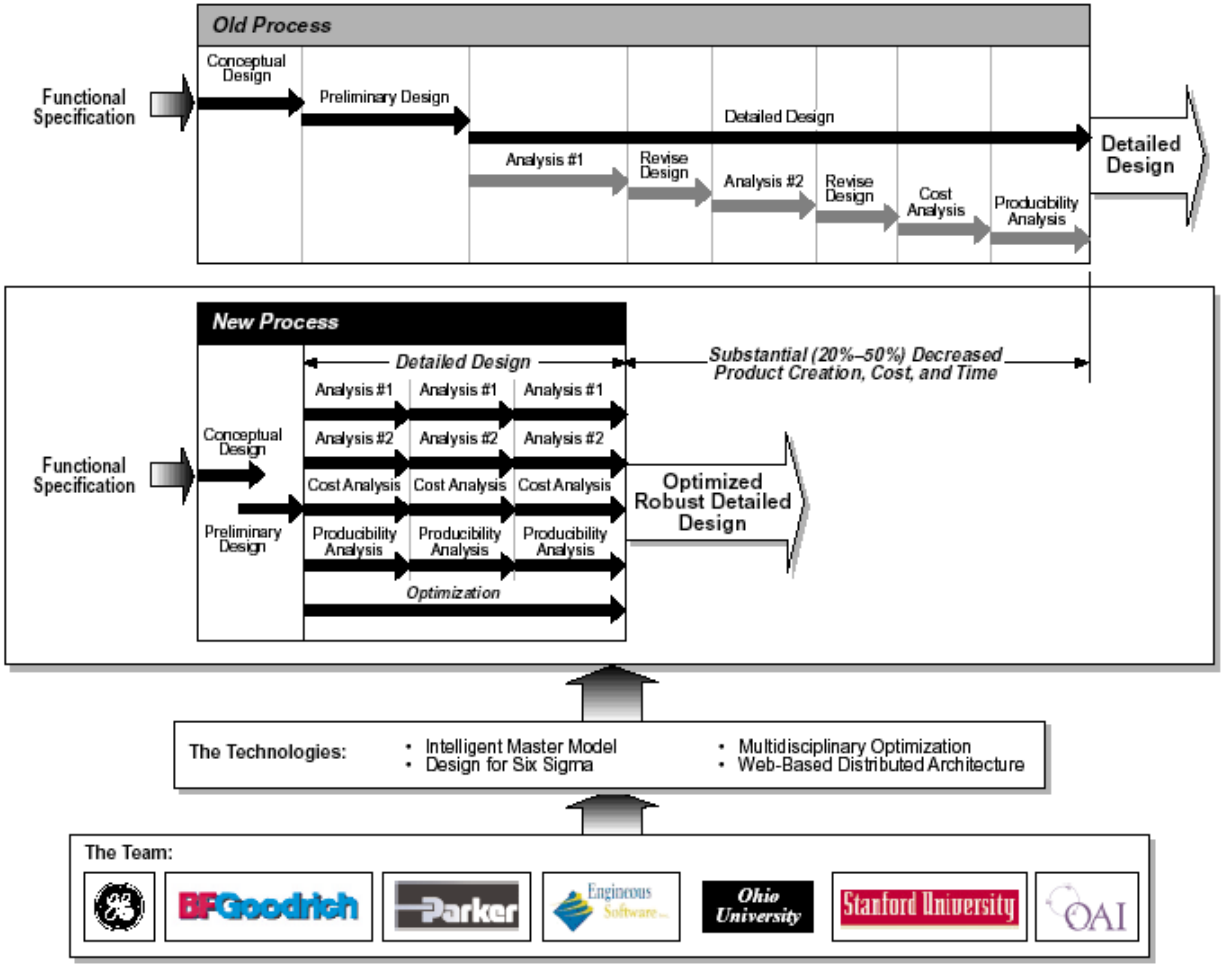
Transdisciplinary (TD CE) Process

Leveraging resources and reuse for R&D growth



Ops: apps, tools, utilities -> **programs**
Metaprogram -> program of **programs**
(process expression by other process expressions)

Federated CCE (FIPER)



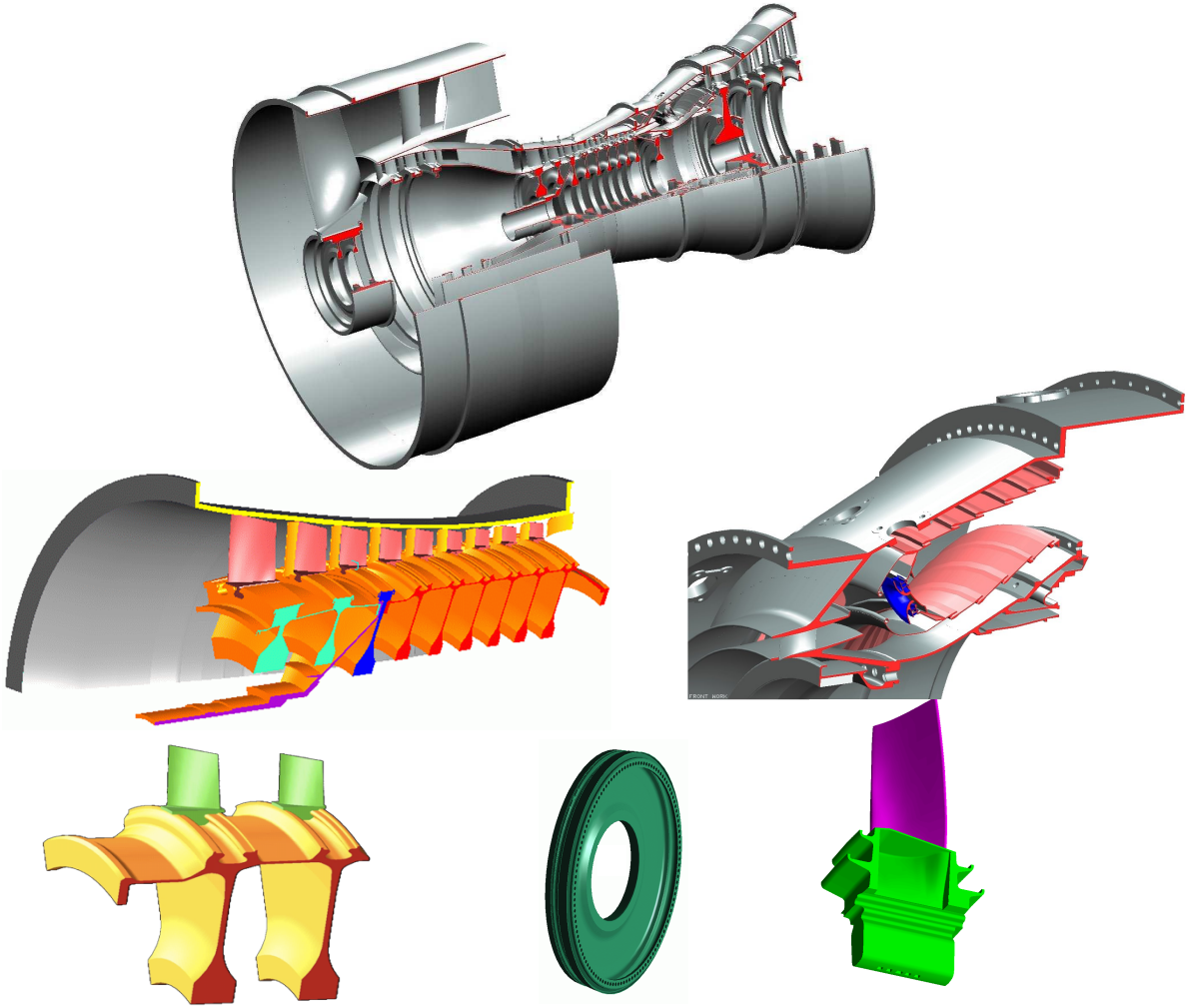
By providing breakthrough product design technology, FIPER will significantly reduce product creation costs and time to market by 20-50%, while improving design robustness. (NIST $21.5 million)

FIPER Metaprogramming Domain

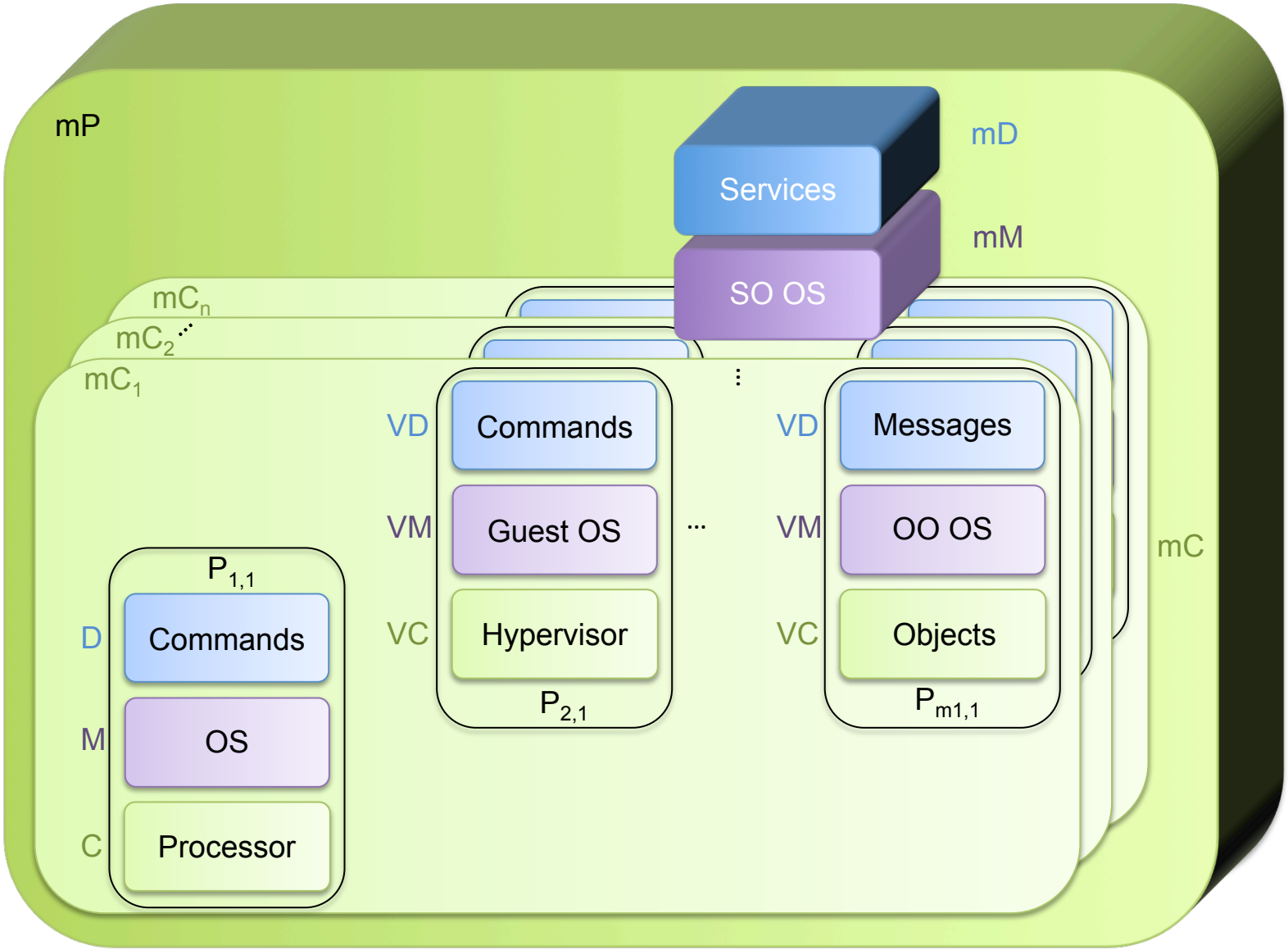
System Design

Subsystem Design

Component Design

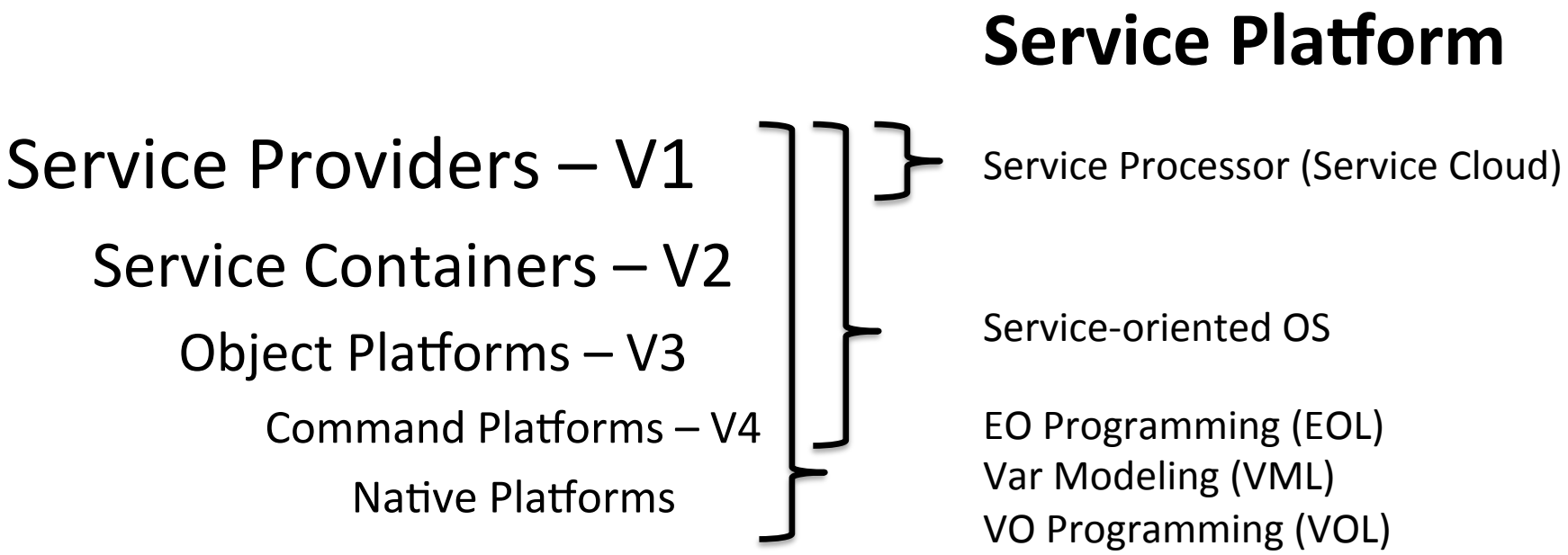


Service-oriented Platform



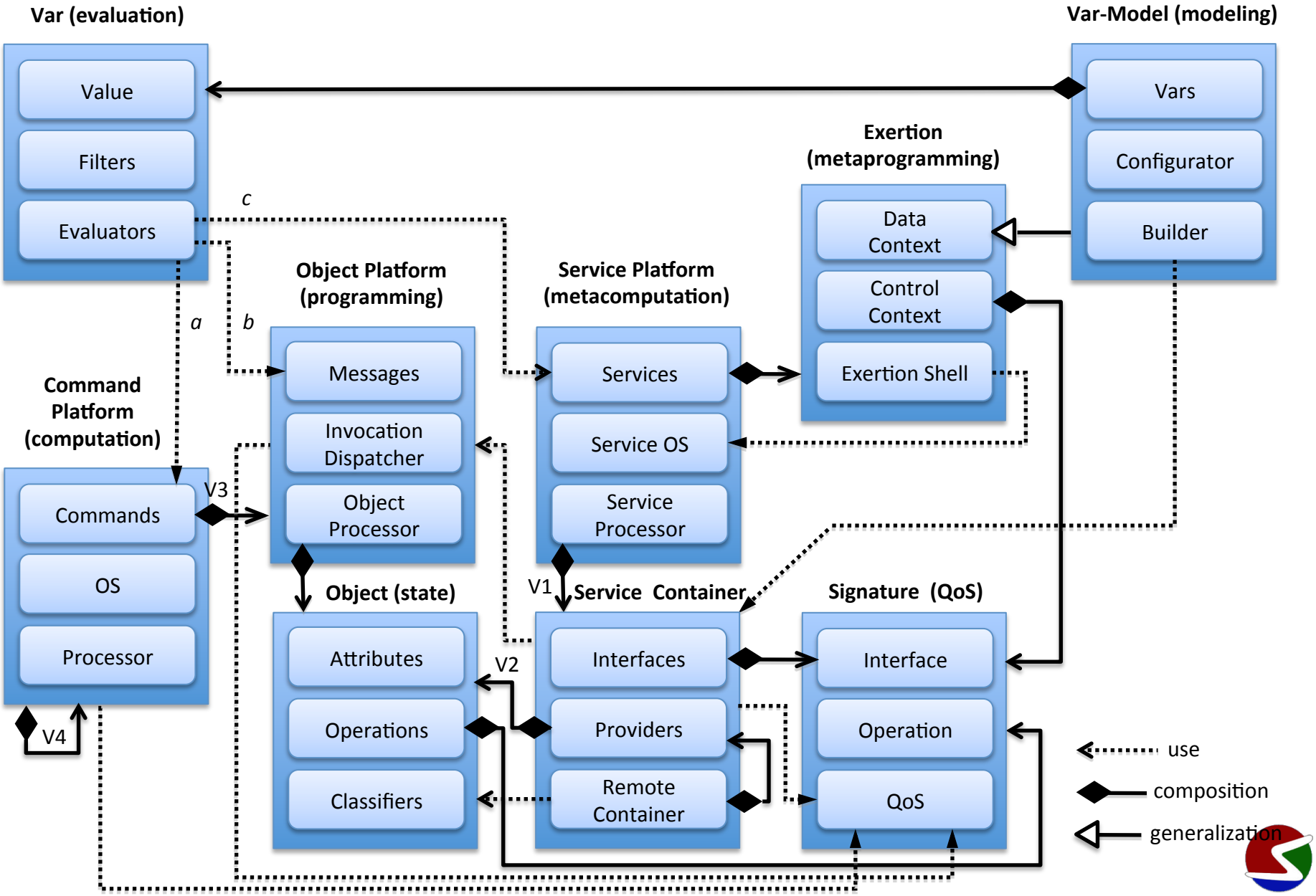
Reducing complexity by higher level abstractions

Virtualization Dependencies

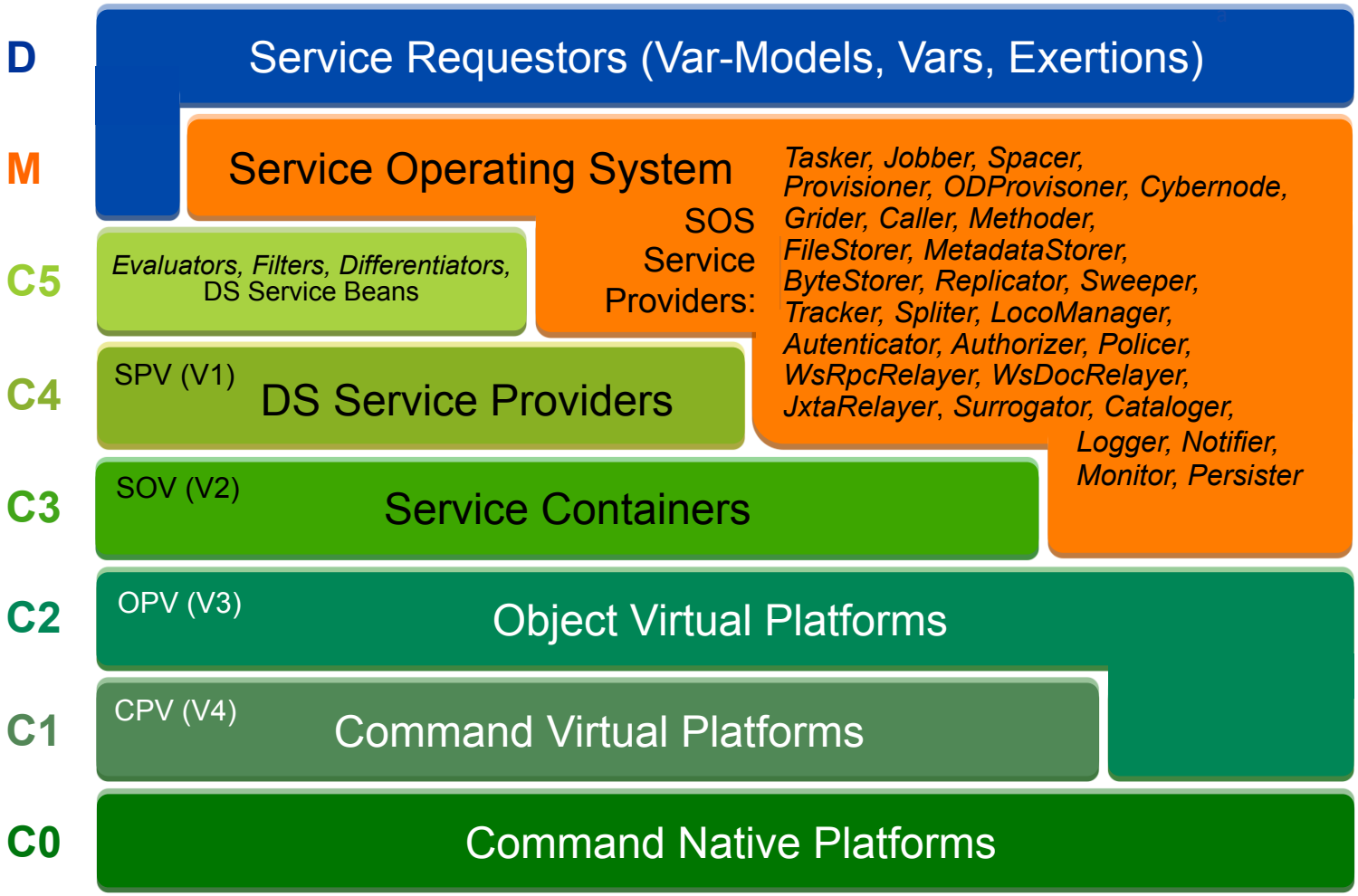


EOL - service collaborations
VOL- multifidelity evaluation compositions
VML - multidisciplinary var-oriented composition

DMC Metamodeling Architecture



SORCER SO Platform



Service requestors (exertions) – commands of the SO processor (C0-C5)

SO program – an exertion executed by the SOS shell

SV – service virtualization, PV – platform virtualization

Agenda

- Intro: computing science & process expression
- Distribution, object & service orientation
- Transdisciplinary computing processes — SO Platform
- C/S, SOA, SPOA, SOOA and **FSOOA**
- SORCER metaprogramming and programming
  - **EOL**, **VOL**, **VML**
- SORCER Operating System (**SOS**) and **FMI**
- SORCER Virtual Processor and **Provisioning**
- Conclusions

Quantum Jumps in Platform Complexity

Sequential Programming
+ order
runtime: batch processing, OS

Multi-threaded programming
- order
+ parallelism
runtime: + concurrency support

Multi-process Programming (time-sharing)
- context
+ SW isolation (safety)
runtime: + interprocess communication (pipes, sockets)

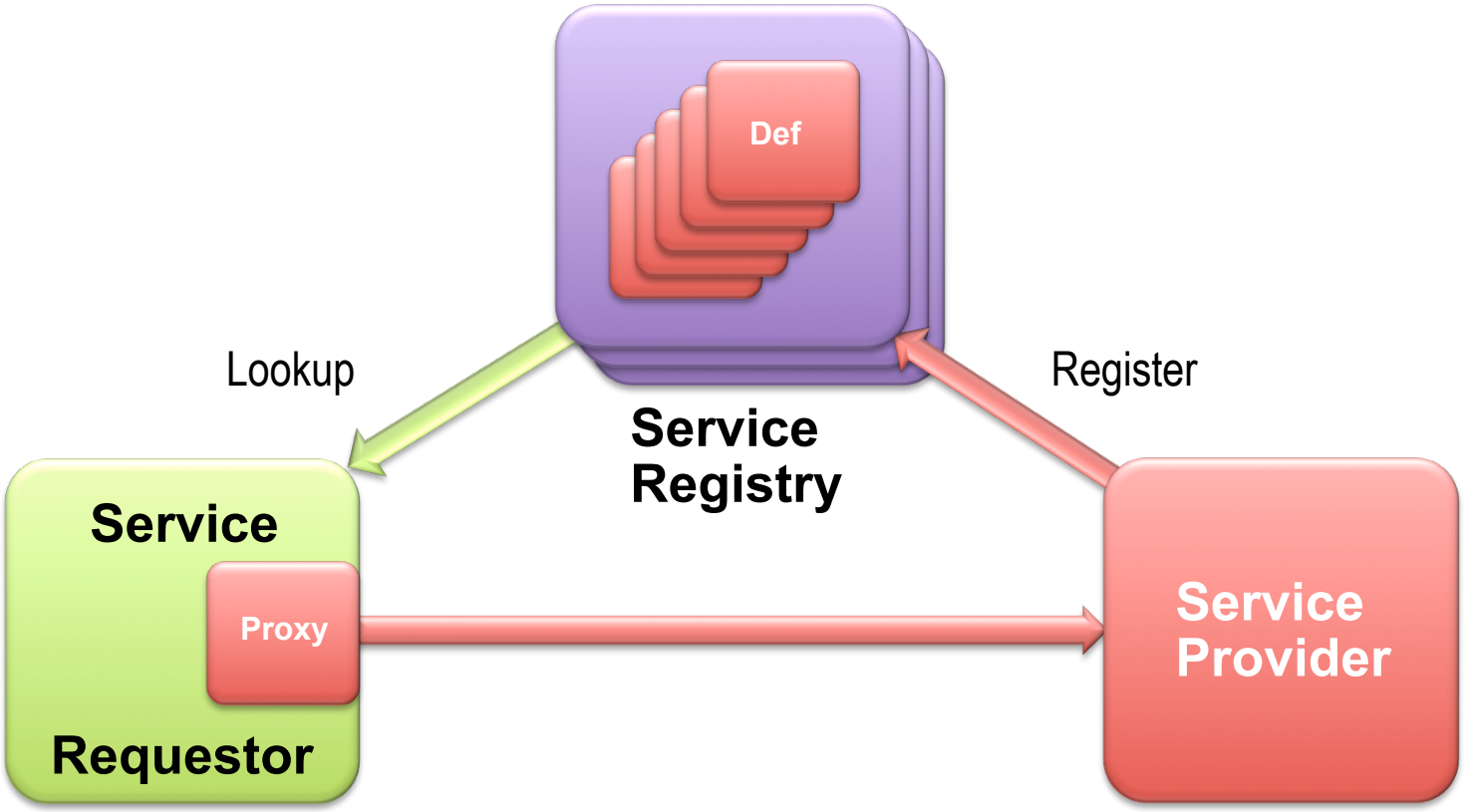
Multi-machine Programming (**client/server**) (DICE, CAMnet, Agile Castings)
- global state, security, trust
+ HW isolation, scalability
runtime: + secure interprocess communication (RPC), trusted mobile code (proxying), virtual file system, disconnected operations, leases, transactions, distributed events, deployment control

Grid Programming (SOA) (FIPER)
- resource setup
+ resource utilization, collocation, distributed resource sharing
runtime: +batch processing (job schedulers) using individual OSs to aggregate CPUs for conventional programs execution

Metaprogramming (FSOOA) (SORCER)
- SO federated programming, DI deployment setup, untrusted mobile code, class loading
+ SO, service federation spontaneity, behavioral transfer, autonomic provisioning
runtime: + service orientation: SO processor, SO OS, and SO programming model

Client/Server

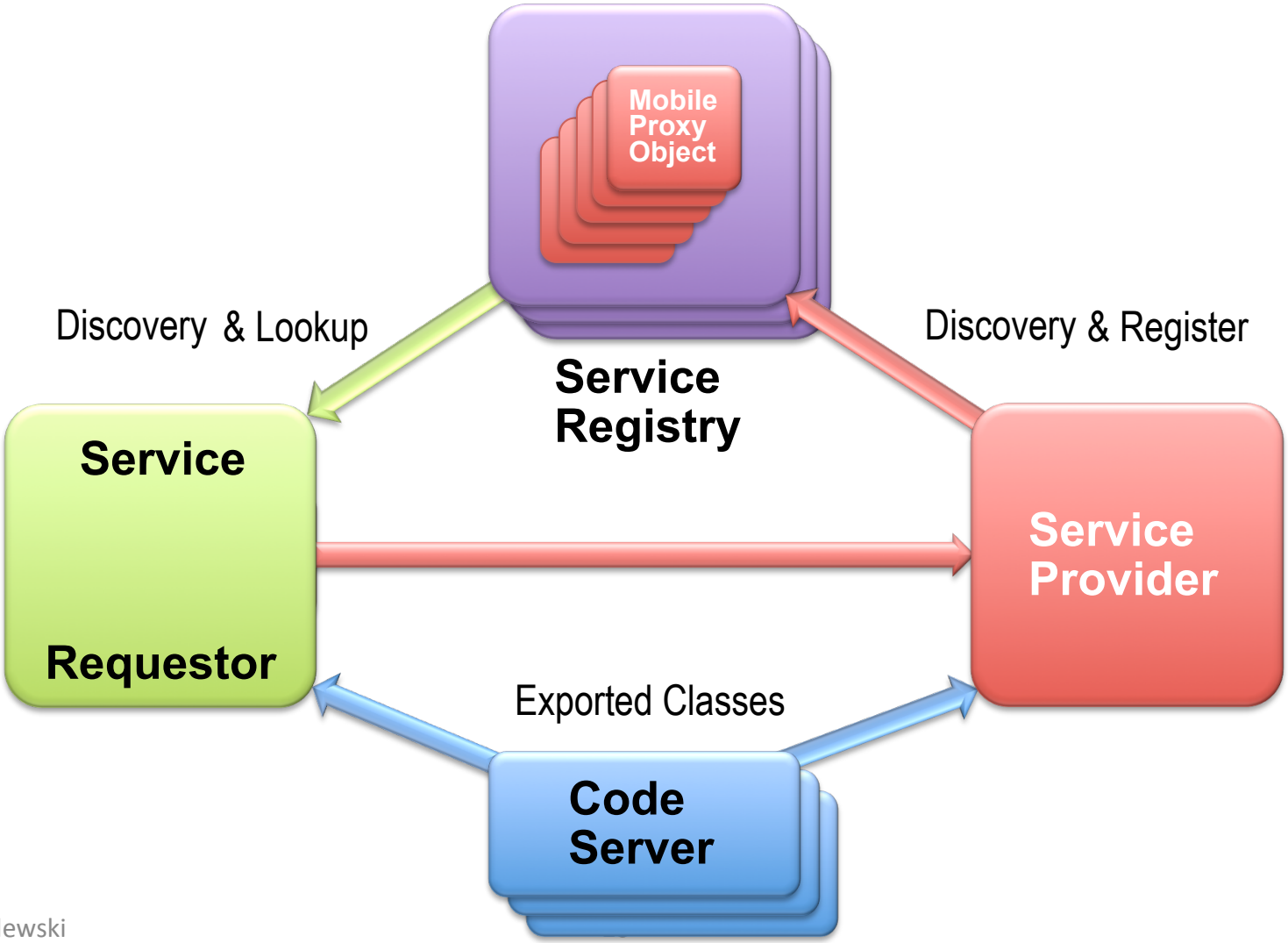
SOA


Def
Lookup
Register
Service
Registry
Service
Proxy
Requestor
Service
Provider

SOOA Terminology

- **Service type** – interface type (service)
- **Service object** – an object implementing its remote service types (services) accessible via its proxy object
- **Service provider** – service object accepting *remote invocations* on one or more its service types
- **Service bean**– local object (POJO) implementing interface types
- **Service container** (service node or cybernode) – service object that *deploys* and *manages* one or more service providers
- **Discovery** – finding out a service registry
- **Lookup** – finding out a service proxy object

SOOA Three Neutralities & BT in SOOA



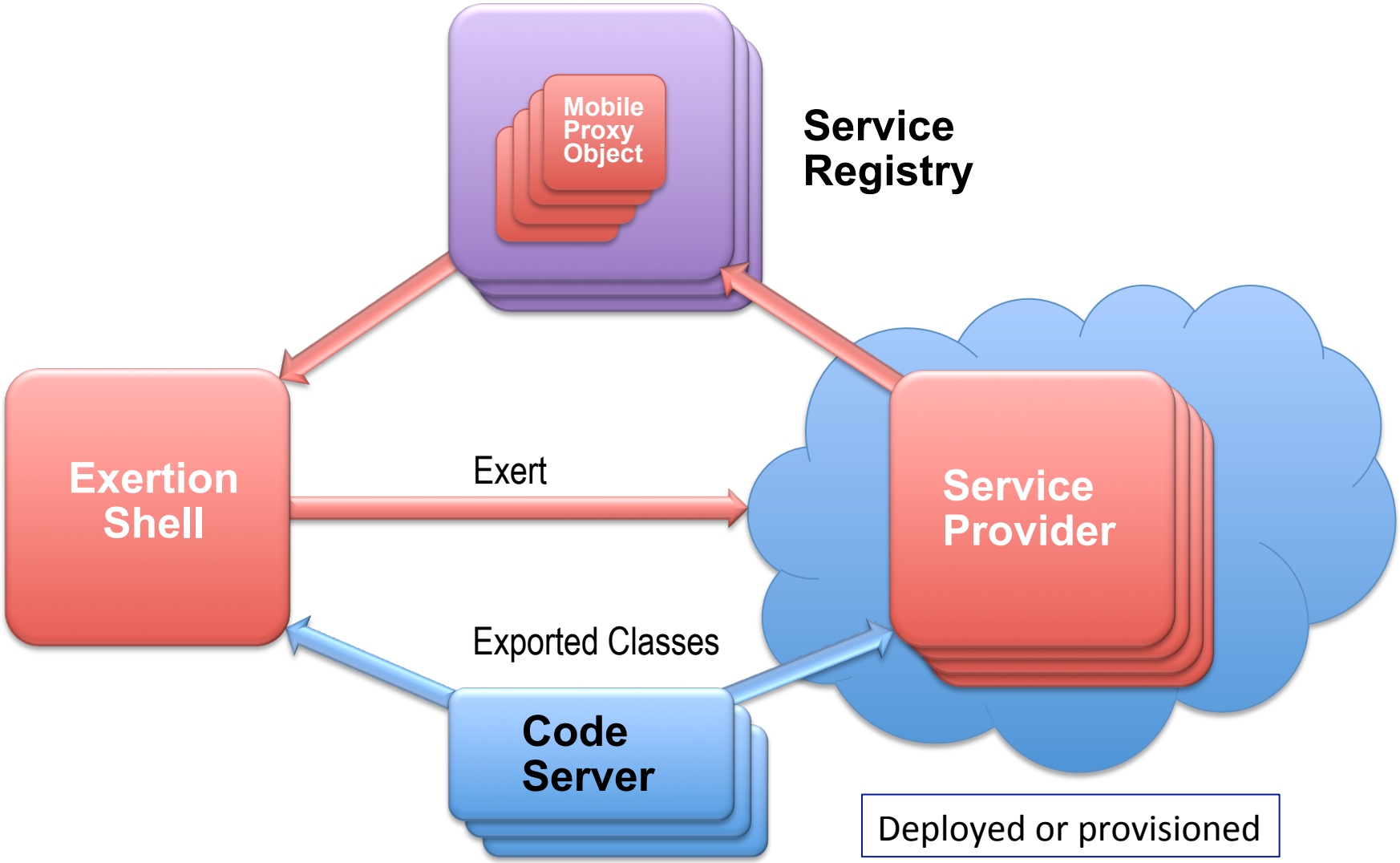
Neutralities

- Implementation
  - Service type (not IDL description)
- Location
  - Dynamic (not static, no endpoints)
- Wire protocol
  - Any (not fixed, e.g. SOAP)
- Data format
  - Generic (*Context* interface) with conversion on external boundaries to XML (no XML within SORCER)

Read-write vs. Remote Invocation
Six RPC Generations

- First generation RPCs: Sun RPC (ONC RPC) , DCE RPC
  - language, architecture, OS independent
  - IDL
- Second generation RPCs: CORBA, Microsoft DCOM-ORPC
  - adds object support
- Third generation: Java RMI
  - it is conceptually similar to the second generation but supports the semantics of object invocation in different address space
  - is built for Java only
  - fits cleanly into the language (interfaces, serialization)
  - no need for standardized data representation
  - with behavioral transfer
- Fourth generation RPCs: Jini Extensible Remote Invocation (Jini ERI)
  - dynamic proxies
  - dynamic configurations (dependency injection
  - security
- Fifth generation RPCs: Web Services RPC and the XML bandwagon
  - SOAP
  - WSDL
- Sixth generation RPCs: SORCER ***Federated Method Invocation*** (FMI)
  - invocation on multiple federating services (virtual metaprocessor)

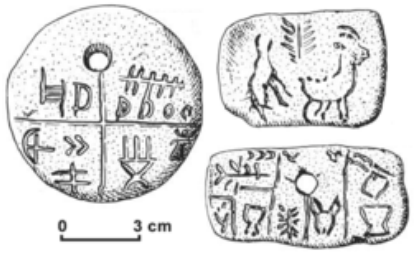
Net-Centric FSOOA



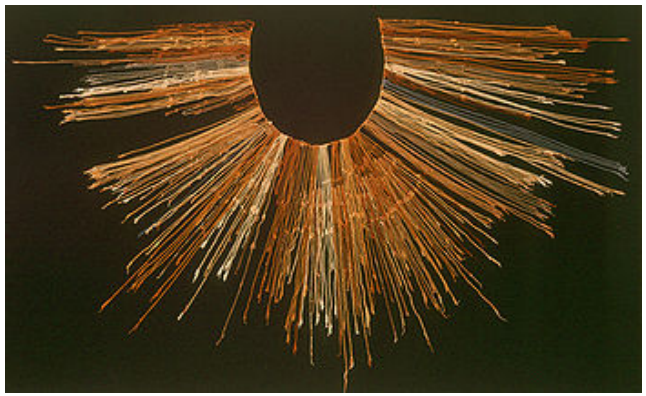
Agenda

- Intro: computing science & process expression
- Distribution, object & service orientation
- Transdisciplinary computing processes — SO Platform
- C/S, SOA, SPOA, SOOA and **FSOOA**
- SORCER metaprogramming and programming
  - **EOL**, **VOL**, **VML**
- SORCER Operating System (**SOS**) and **FMI**
- SORCER Virtual Processor and **Provisioning**
- Conclusions

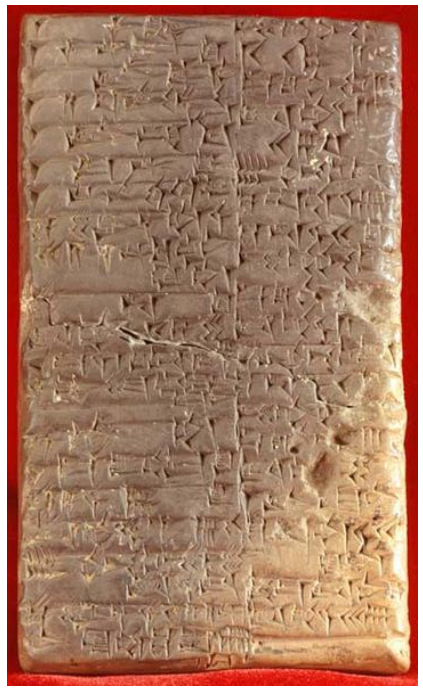
Language – Mankind
Writing – Civilization



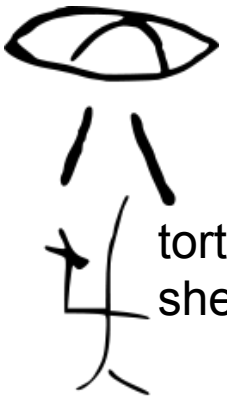
Tărtăria
tablets



quipu



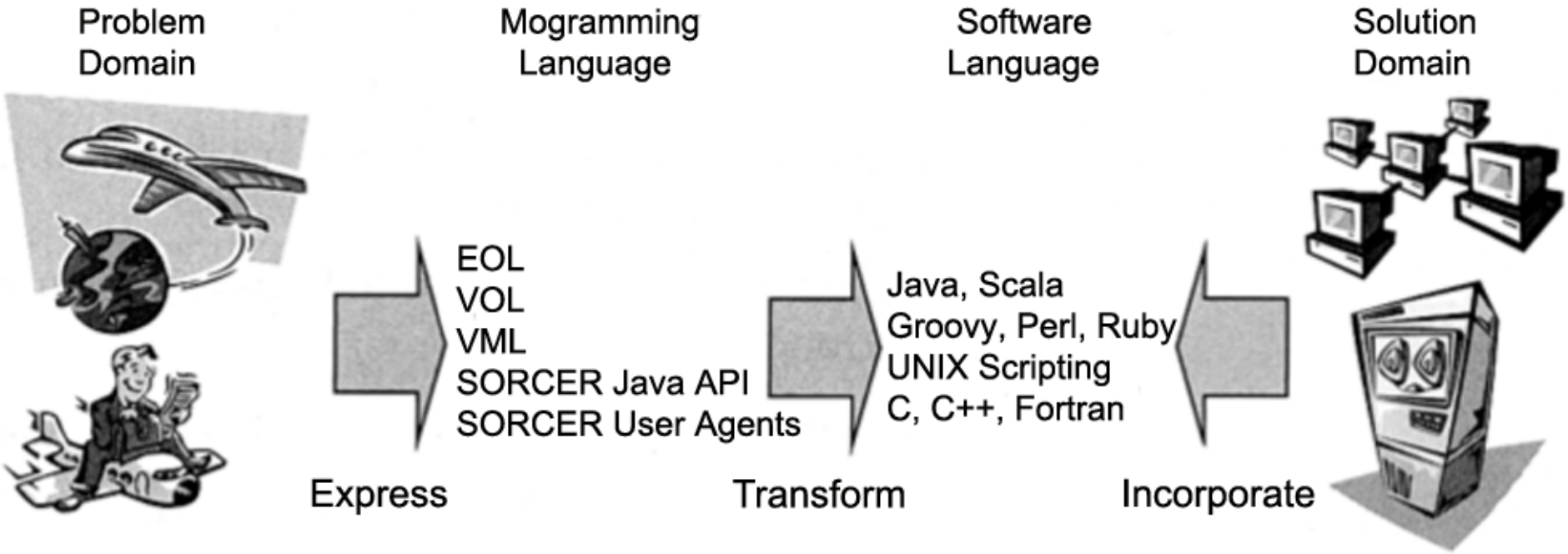
cuneiform
script



tortoise
shells

movable metal type, and composing stick,
descended from Gutenberg's press

Language Eng. vs. SW Eng.



Language engineering is the art of creating languages.

Requestor Metaprogramming Abstractions

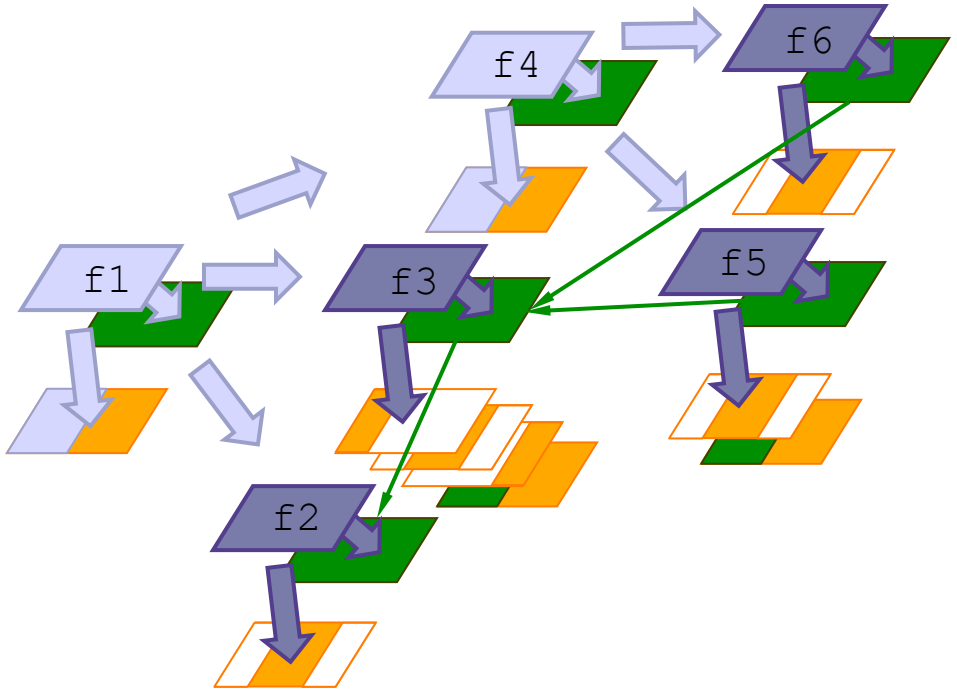
- EO Programming
  - Service collaborations
- VO Programming
  - Active variables (vars) composition
- VO Modeling
  - Model-driven VOP for multi-fidelity, multi-scale, multi-disciplinary collaborations

scripting, Java API, hybrid, visual

Service Composition

```
f = f1(f2, f3, f4(f5, f6)
```

Meta Program (Exertion)



Signature type: preprocess process postprocess append

Task: network instruction

```
task(
    sig("multiply", Multiplier.class),
    context(
        input("arg/x1”, 10.0d),
        input("arg/x2”, 50.0d)))
```

Job: service composition

f1(f2(f4, f5), f3)

```
Task f4 = task("f4", op("multiply", Multiplier.class),
    context("multiply", input("arg/x1”, 10.0d),
        input("arg/x2”, 50.0d), out("result/y1”, null)));

Task f5 = task("f5", op("add", Adder.class),
    context("add", input("arg/x3”, 20.0d),
        input("arg/x4”, 80.0d), output("result/y2”, null)));

Task f3 = task("f3", op("subtract", Subtractor.class),
    context("subtract", input("arg/x5”, null),
        input("arg/x6”, null), output("result/y3”, null)));

Job f1= job("f1", job("f2", f4, f5,
    strategy(Flow.PAR, Access.PULL)), f3,
    pipe(output(f4, "result/y1”), input(f3, "arg/x5”)),
    pipe(output(f5, "result/y2”), input(f3, "arg/x6")));
```

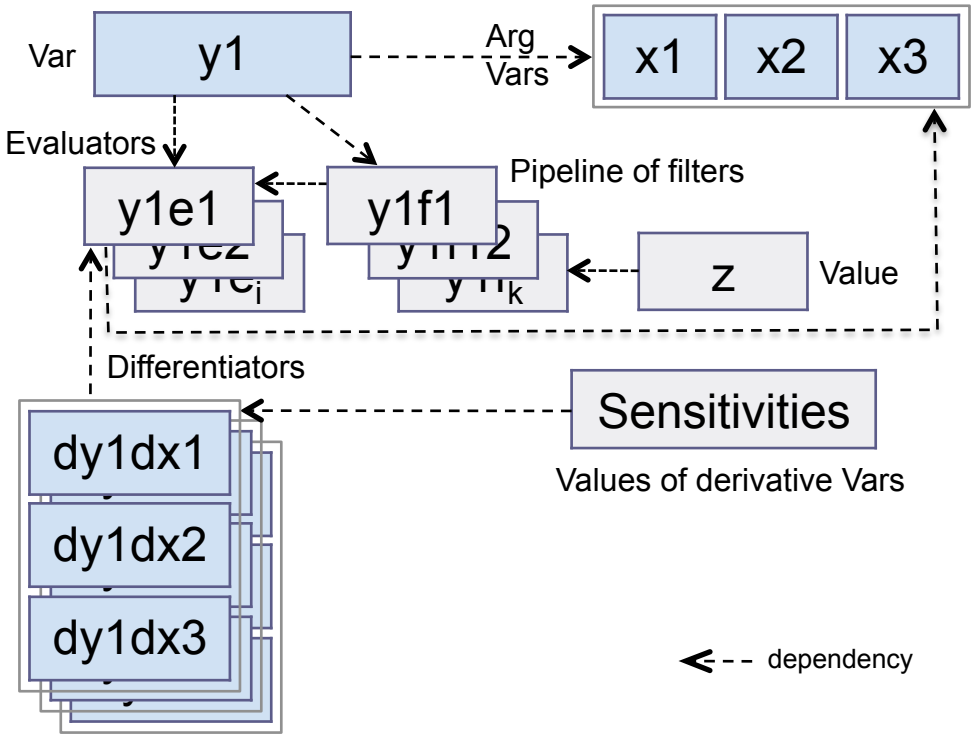
Can use Control Flow Exertions

Types of Variables

- Variable (mathematics), a symbol that represents a quantity in a mathematical expression
- Variable (programming), a symbolic name associated with a value that may be changed
- Variable (OO programming), a set of object's attributes accessible via 'getters'
- Variable (SO programming), a triplet <value, evaluator, filter>
  - value: a valid quantity
  - evaluator: a service with dependent variables (composition)
  - filter: a getter

Basic Variable Structure (VFE)

$$z = y_1(x_1, x_2, x_3)$$



Service Orientation

- A **service**: the work performed by a variable's evaluator
- An **evaluator** defines:
  - Arguments (variable composition)
  - Processing services (mutifidelity)
  - Differentiation services (mutifidelity)

Type of Evaluator Services

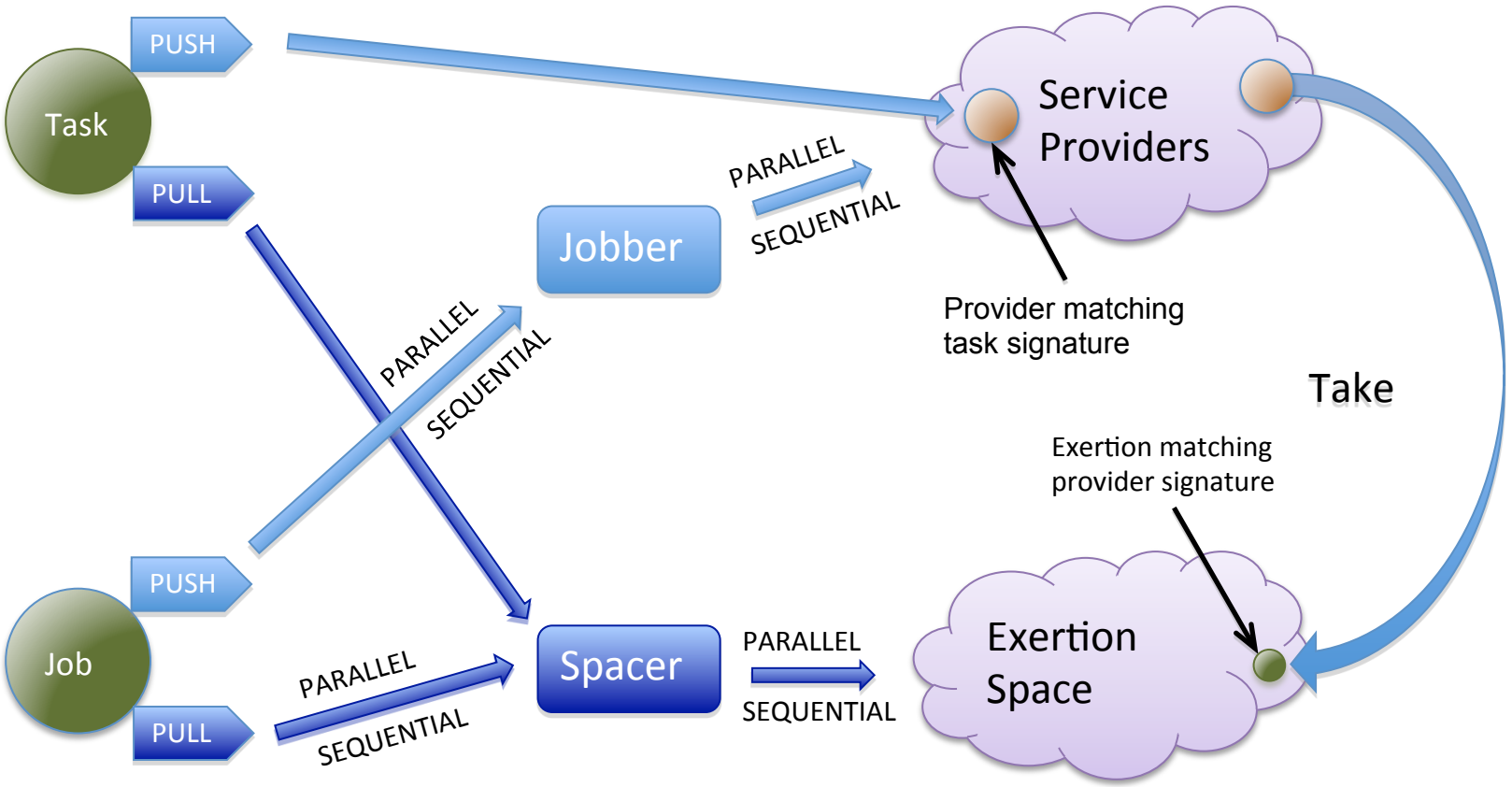
- Command services (a)
  - Execute command (executables)
- Scripting services (a)
  - Execute expression (e.g., scripts, Java expressions)
- OO services (b)
  - Method invocation, RMI
- Federated services (c)
  - Federated method invocation (exertions)
    - Execute remote command
    - Execute expression remotely
    - Remote method invocation
    - Federated method invocation

Evaluators do realize other process expressions!—enable metaprogramming

Agenda

- Intro: computing science & process expression
- Distribution, object & service orientation
- Transdisciplinary computing processes — SO Platform
- C/S, SOA, SPOA, SOOA and **FSOOA**
- SORCER metaprogramming and programming
  - **EOL**, **VOL**, **VML**
- SORCER Operating System (**SOS**) and **FMI**
- SORCER Virtual Processor and **Provisioning**
- Conclusions

Push vs. Pull execution



Pull execution allows for a pandemonium SO architecture.

Applying OO to Network (FMI)

- Service request is an object of type:
  `Exertion` = Data **`Context`** + **`Signature`**s
- Exertions are invoked by calling **`exert:`**
  `Exertion#exert(Transaction):Exertion`
- Exertions are executed by the network shell using *collaborating service providers* of type: **`Servicer`**
  - Service providers form P2P (S2S) environment
  - Service is requested by calling dynamically the **`service`** method
    `Servicer#service(Exertion, Transaction):Exertion`
  - A service provider is identified by the exertion signature's **interface type** and optional attributes
- The signature operation <**`operation`**> is invoked on the matching *service object*:
  `public Context <operation>(Context)` via
  `Exerter#exert(Exertion, Transaction):Exertion`

UNIX Platform vs. SORCER Platform

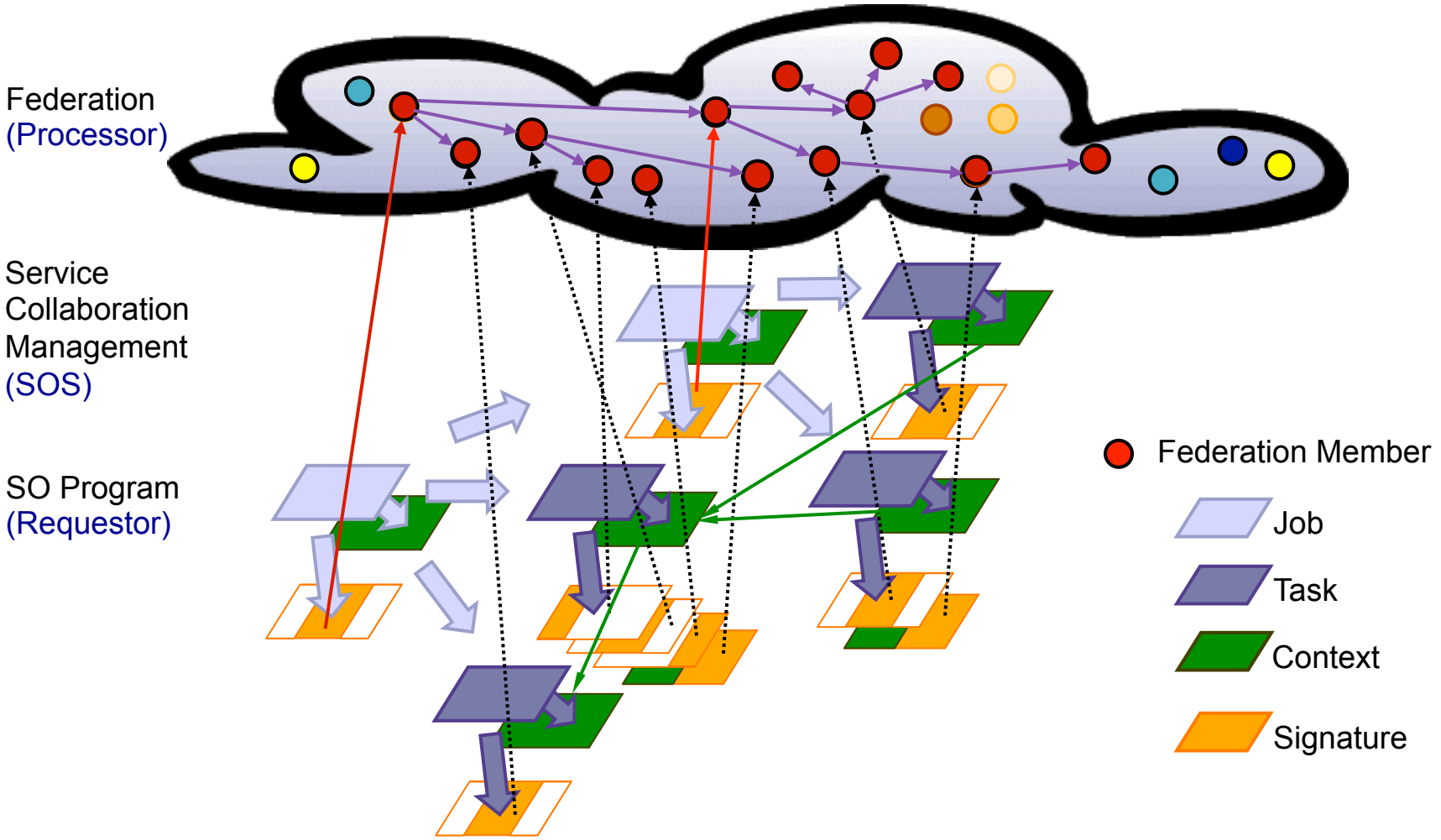
	UNIX	SORCER
Data	File - file system	Data context - objects
Data flow	Pipes	Data context pipes
Cohesion	Everything is a file	Everything is a service
Processor	Native	Service providers
Interpreter	Shell	Exertion (network) shell
System language	C	Java/Jini/Rio/SORCER API
Command language	UNIX shell scripting	EO/VO/MD scripting
Process control strategy	Command flow logic	Control context & exertion flow logic (looping and branching)
Executable codes	Many choices	Many choices

Unix pipes – processes; SORCER pipes – data contexts
Pipeline vs. SORCER federation – exertion + control context + control flow exertions
Local shell vs. network shell

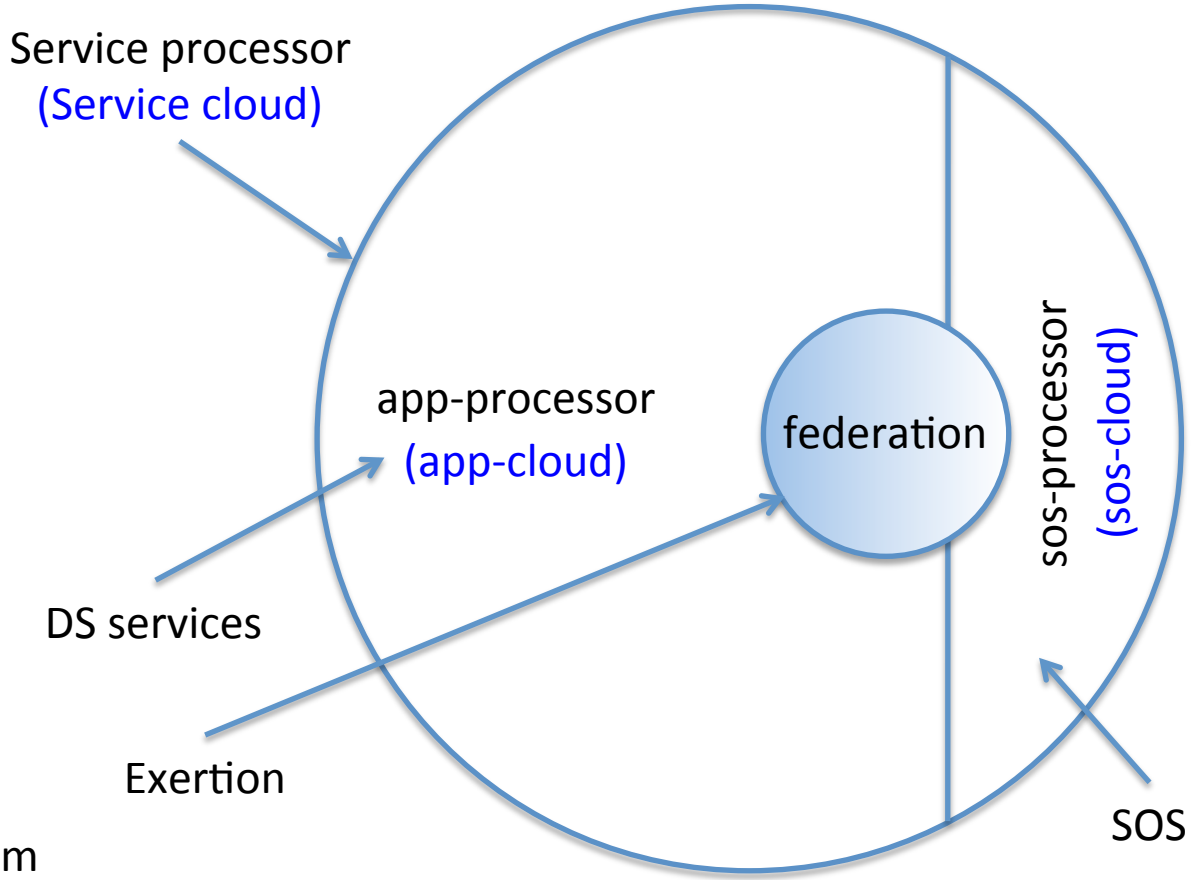
Agenda

- Intro: computing science & process expression
- Distribution, object & service orientation
- Transdisciplinary computing processes — SO Platform
- C/S, SOA, SPOA, SOOA and **FSOOA**
- SORCER metaprogramming and programming
  - **EOL**, **VOL**, **VML**
- SORCER Operating System (**SOS**) and **FMI**
- SORCER Virtual Processor and **Provisioning**
- Conclusions

Exerting Dynamic Collaborations



SORCER Platform: Services



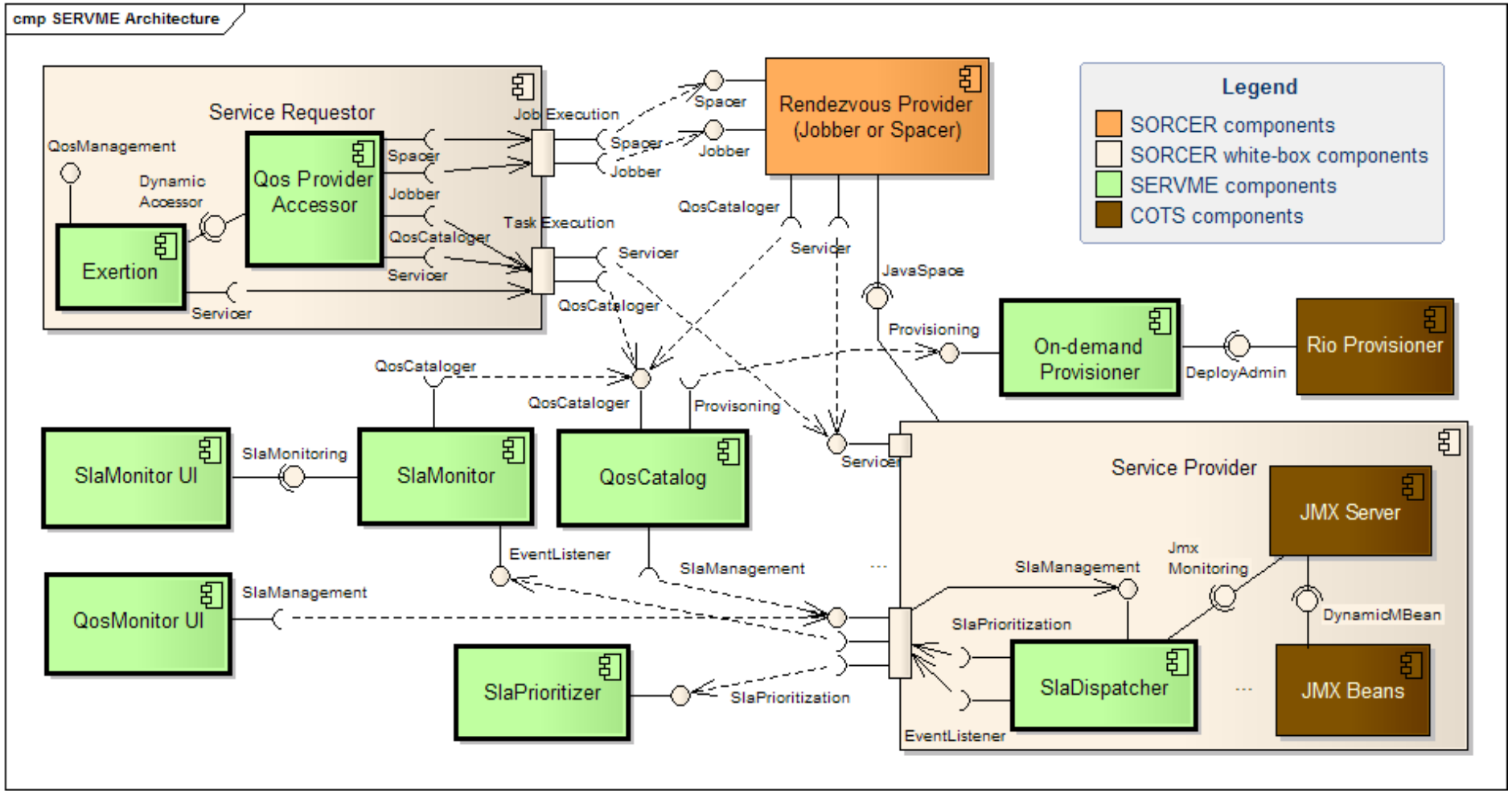
Native Platform
- Command Platform (V)
  - Object Platform (V)
    - Service Node (V)
    - Cybernode (V)
      - Service Provider (P)

Blue indicates services on VCP

QoS and SLA

- **Quality of Service (QoS) Parameter**
  a technical characteristic or performance benchmark of a resource
- **Service Level Agreement (SLA)**
  a contract signed between a service requestor and a service provider for a specific time or task. It specifies that during the execution certain QoS parameters should maintain an agreed-on level or a fixed value.

Provisioning Component Diagram



Provisioning Types

Type	Bootstrapping	Monitoring	Timeline
manual	manual	no	till destroyed
manual-dynamic	manual	yes	till destroyed/configured
autonomic	auto	yes	till destroyed/configured
on-demand	auto	no	configured
on-demand-dynamic	auto	yes	till destroyed/configured

Agenda

- Intro: computing science & process expression
- Distribution, object & service orientation
- Transdisciplinary computing processes — SO Platform
- C/S, SOA, SPOA, SOOA and **FSOOA**
- SORCER metaprogramming and programming
  - **EOL**, **VOL**, **VML**
- SORCER Operating System (**SOS**) and **FMI**
- SORCER Virtual Processor and **Provisioning**
- Conclusions

Conclusions
SORCER FSOOA

- Discovery/join protocols
  - Location neutrality
- Service provider registration
  - Proxy object implementing service types
  - Proxy object owned by the provider
  - Proxy wire protocol(s) selected by provider
- Light-weight containers (service node, cybernode)
  - Small footprint JVM Hosting service providers
  - Static or dynamic deployment of service providers
  - Service assembly by DI
- OS (Tasker, Jobber, Spacer, Cataloger, Provisioner, Cybernode, …)
  - FMI
  - Synchronous, asynchronous, QoS-optimized service federations
  - **Provisioning**
  - **V1-V4 Enables Two Way Computing Convergence**
  - SO shell

Q&A